Prüfer: Prof. Dr. Kurt Rothermel

Betreuer: Dr. Cora Burger
Dipl.-Inform. Hartmut Benz

Begonnen am: 01.06.1996

Beendet am: 30.11.1996

CR-Nummer: C.2.4, C.2.M, D.2.1, H.5.2, H.5.3

Diplomarbeit Nr.1421

Benutzeragenten zur Unterstützung von Teamkoordination

Jörg Zimmer

Kurzfassung

Die Unterstützung der Teamarbeit ist, im Zuge der zunehmenden räumlichen Verteilung der einzelnen Mitarbeiter eines Teams, eine der wichtigsten Aufgaben im Gebiet der verteilten Systeme. Die Teamarbeit in solchen Gruppen besteht aus einer großen Anzahl von Aktivitäten der einzelnen Mitarbeiter, die, um Chaos und Ineffizienz zu vermeiden, aufeinander abgestimmt werden müssen.

Um dies zu leisten, sollen die einzelnen Mitarbeiter die Möglichkeit haben, sich über die momentane oder zukünftige K ooperationsbereitschaft ihrer Kollegen zu informieren.

Im Rahmen dieser Diplomarbeit wurde nach eingehender Untersuchung vorhandener Koordinations- und Kooperationswerkzeuge aus dem Bereich des CSCW festgestellt, daß zur Zeit noch keine Koordinationswerkzeuge am Markt existieren, die alle der folgenden Kriterien e rfüllen:
Es soll die Möglichkeit bestehen, alle zur Kooperation benötigten Informationen zu verwalten. Verschiedene Kommunikations- und Koordinationswerkzeuge sollten integriert werden können, die Kooperationsbereitschaft eines Mitarbeiters in einem Team sollte direkt am Bildschirm angezeigt werden und das Werkzeug sollte auf einem Agentensystem mit mobilen Agenten au fbauen.

Daraufhin wurde das die Teamkoordination unterstützende Werkzeug Mole-Office aus der Sicht verschiedener Untersuchungsstandpunkte spezifiziert und mittels dem USCM-Modell entwo rfen.

Anschließend wurde eine prototypische Implementierung des Softwaresystems in der Programmiersprache Java, unter der Verwendung des Agentensystems Mole, vol lzogen.

Inhaltsverzeichnis

1. EINLEITUNG 6

1.1 AUFGABENSTELLUNG 6

1.2 MOTIVATION 6

1.3 ÜBERSICHT 7

2. COMPUTERUNTERSTÜTZTES KOOPERATIVES ARBEITEN (CSCW) 8

2.1 CSCW-GRUNDLAGEN 8

2.1.1 Die Kommunikation 8
2.1.2 Die Koordination 9
2.1.3 Die Kooperation 10

2.2 EIN GRUNDLEGENDES KOOPERATIONSMODELL 11

2.2.1 Aktivitäten, Orte und Mitarbeiter 11
2.2.2 Das Kooperationsmodell 14

2.3 KLASSIFIZIERUNG VON KOORDINATIONS- UND KOOPERATIONSWERKZEUGEN 16

2.3.1 Klassen von Koordinationswerkzeugen 16
2.3.2 Klassen von Kooperationswerkzeugen 19

2.4 ANALYSE BESTEHENDER KOORDINATIONS- UND KOOPERATIONSWERKZEUGE 20

2.4.1 Die Funktionalität vorhandener Koordinationswerkzeuge 20
2.4.1.1 Calendar Manager CM 20
2.4.1.2 Microsoft Schedule+ 21
2.4.1.3 CAP II 22
2.4.1.4 Montage 23
2.4.2 Die Funktionalität vorhandener Kooperationswerkzeuge 24
2.4.2.1 DIVA 24
2.4.2.2 GroupDesk System 25

3. AGENTENTECHNOLOGIE 27

3.1 DEFINITIONEN 27

3.2 AGENTENSYSTEME MIT MOBILEN AGENTEN 32

3.2.1 Java-To-Go 32
3.2.2 CyberAgent 33
3.2.3 Ara 35
3.2.4 Telescript 36
3.2.5 Mole 37

4. PRODUKTSPEZIFIKATION 38

4.1. DER UNTERNEHMENSSTANDPUNKT 39

4.2 DER INFORMATIONSSTANDPUNKT 41

4.3 DER BERECHNUNGSSTANDPUNKT 43

4.3.1 Die Aspekte der Benutzerschnittstelle 43
4.3.2 Die Aspekte der Schnittstelle zum Mailtool 44
4.3.3 Die Aspekte der Schnittstelle zum Kalendermanagementwerkzeug CM 44

4.3.4 Die Aspekte der Funktionalität von Mole-Office 45
4.4 DER ENGINEERINGSTANDPUNKT 46
4.5 DER TECHNOLOGIESTANDPUNKT 47
5. ENTWURF DES SOFTWARESYSTEMS MOLE-OFFICE 49
5.1 DER SUBSTANTIELLE BEREICH 50
5.2 DER GEBRAUCHSBEREICH 60
5.3 DER VERWALTUNGSBEREICH 66
5.4 DER SYSTEMKERN 70
6. IMPLEMENTIERUNG 79
6.1 DIE ARCHITEKTUR DES MOLE-OFFICE-SYSTEMS 79
6.2 DIE BENUTZEROBERFLÄCHE VON MOLE-OFFICE 80
6.3 MODELLIERTE OBJEKTE IM MOLE-OFFICE-SYSTEM 86
*6.3.1 Der Administrator-Agent (**MOAdministratorAgent**) 86*
*6.3.2 Der System-Agent (**MSSystemAgent**) 88*
*6.3.3 Der Ask-Agent (**MSAskAgent**) 90*
*6.3.4 Der Watch-Agent (**MSWatchAgent**) 91*
*6.3.5 Das Benutzer-Objekt (**User**) 92*
*6.3.6 Das Raum-Objekt (**Room**) 93*
7. ZUSAMMENFASSUNG 94
7.1 ERGEBNISSE 94
7.2 AUSBLICK 95
8. ANHANG 96
8.1 INSTALLATION UND START VON MOLE-OFFICE 96
8.2 DIE MOMENTANE BIBLIOTHEK VON MOLE-OFFICE 98
8.2.1 Modellierte Objekte im Mole-Office-System 98
8.2.2 Die Benutzeroberfläche von Mole-Office 104
8.3 LITERATURVERZEICHNIS 113
8.4 ERKLÄRUNG 119

Abkürzungsverzeichnis

CM..... Suns Calendar Manager

CSCW..... Computer Supported Cooperative Work

E-Mail..... elektronische Mail

ICAP..... Internet Calendar Access Protocol

IMAP..... Internet Mail Access Protocol

IPVR..... Institut für paralle und verteilte Höchstleistungsrec hner

SMTP ..... Simple Mail Transfer Protocol

USCM..... Universal Service Component Model

1. Einleitung

1.1 Aufgabenstellung

Um die Erreichbarkeit für Ansprechpartner innerhalb einer Arbeitsgruppe zu erhöhen, heften manche Mitarbeiter beim Verlassen ihres Büros Zettel an ihre Türen, die ihren momentanen Aufenthaltsort und eventuell die geplante Aufenthaltsdauer angeben. Im Zuge der zunehmenden räumlichen Verteilung von Arbeitsplätzen und der zunehmenden Unterstützung der Mitarbeiter durch vernetzte Arbeitsumgebungen, erscheint es wünschenswert, dieses Verhalten direkt vom Arbeitsplatz aus elektronisch nachahmen zu können und sogenannte persönliche Softwareagenten dafür einzusetzen.

Im Rahmen dieser Diplomarbeit soll ein Softwaresystem entworfen und implementiert werden, das auf den Arbeitsplatzrechnern der Mitarbeiter der Abteilung IPVR der Fakultät Informatik der Universität Stuttgart lauffähig ist und das solche elektronischen Zettel realisiert, d.h. auf Anfrage von anderen Benutzern die momentane Erreichbarkeit ihrer Benutzer bekanntgibt.

Dazu wird einerseits eine graphische Oberfläche benötigt, mit der die Mitarbeiter ihre Aufenthaltsorte außerhalb des Büros, oder besondere nicht zu unterbrechende Tätigkeiten definieren und die Erreichbarkeit anderer abfragen können. Andererseits muß das Softwaresystem in ein Agentensystem integriert werden, damit ein persönlicher Softwareagent realisiert werden kann. Durch die Integration von Email soll Anrufbeantworterfunktionalität ermöglicht werden. Zusätzlich wird ein Kalendermanagementwerkzeug integriert, damit bei fest eingetragenen Terminen automatisch angegeben werden kann, wo und wie lange sich der Mitarbeiter an einem bestimmten Aufenthaltsort befindet. Außerdem sollen verschiedene Vertraulichkeitsstufen definiert und beachtet werden.

1.2 Motivation

Die Hauptbeweggründe für diese Diplomarbeit sind die Verbesserung der Teamarbeit durch Erhöhung der Erreichbarkeit einzelner Mitarbeiter unter Berücksichtigung ihrer Privatssphäre. Die Teamarbeit wird im weiteren verbessert durch Verstärkung des Teamgedankens zwischen den Mitarbeitern, da das Team als Ganzes, mit seinen Mitarbeitern und deren Aufenthaltsorten, dargestellt werden kann und durch Zeitersparnis einzelner Mitarbeiter, da alle Anfragen im Bezug auf andere Mitarbeiter direkt vom Arbeitsplatzrechner aus durchgeführt werden können. Durch die Verwendung von mobilen Agenten soll eine effiziente Arbeitsumgebung geschaffen werden, die im Bereich von verteiltem Arbeiten immer wichtiger wird, wobei hier die Idee eines persönlichen Softwareagenten für jeden Mitarbeiter im Hintergrund steht. Der persönliche Agent eines Mitarbeiters soll

für den ihn kleine Routinearbeiten übernehmen und somit die alltägliche Arbeit und Zusammenarbeit erleichtern und beschleunigen.
Desweiteren sollen Email und SUNs Calendermanager CM, also verschiedene Kommunikations- und Koordinationswerkzeuge, die einzelne Funktionen in der Teamkoordination übernehmen können, in das Softwaresystem integriert werden. Dadurch wird ein integrierender Ansatz realisiert, so daß am Ende der Entwicklung des Softwaresystems **ein Werkzeug** zur Unterstützung der Teamkoordination vorhanden ist, das alle verfügbaren Werkzeuge miteinander verbindet.

1.3 Übersicht

Diese Diplomarbeit gliedert sich in sieben Teile. Nach dem Einleitungsteil, werden anschließend im zweiten und dritten Abschnitt wichtige Teilgebiete der Diplomarbeit, wie z.B. verschiedene Werkzeuge des Computer Supported Cooperative Work (CSCW) und die Agententechnologie daraufhin untersucht, ob vergleichbare Ansätze bereits existieren und ob die verschiedenen Werkzeuge entsprechend geeignet sind.

Im vierten Teil, dem Spezifikationsteil, werden diverse Konzepte, Ansätze und Entwürfe erläutert, deren Entwurf und Implementierung im fünften und sechsten Teil beschrieben werden.

Im abschließenden siebten Teil werden die Ergebnisse zusammengefaßt und die Möglichkeiten für die zukünftige Erweiterbarkeit der Anwendung aufgezeigt.

2. Computerunterstütztes Kooperatives Arbeiten (CSCW)

Da sich das Gebiet des CSCW[1] mit dem für diese Diplomarbeit zentralen Thema der Teamkoordination beschäftigt, werden in diesem Kapitel zunächst die grundlegenden Begriffe und Konzepte des CSCW näher erläutert. Anschließend werden die Werkzeuge entsprechender relevanter Teilbereiche von CSCW klassifiziert und ausgewählte Anwendungen daraufhin überprüft, ob bereits Werkzeuge existieren, die den Anforderungen der Diplomarbeit zur Unterstützung von Teamkoordination genügen.

2.1 CSCW-Grundlagen

Ein CSCW-System baut, wie in folgender Abbildung veranschaulicht, auf Kommunikation, Koordination und Kooperation auf[2]:

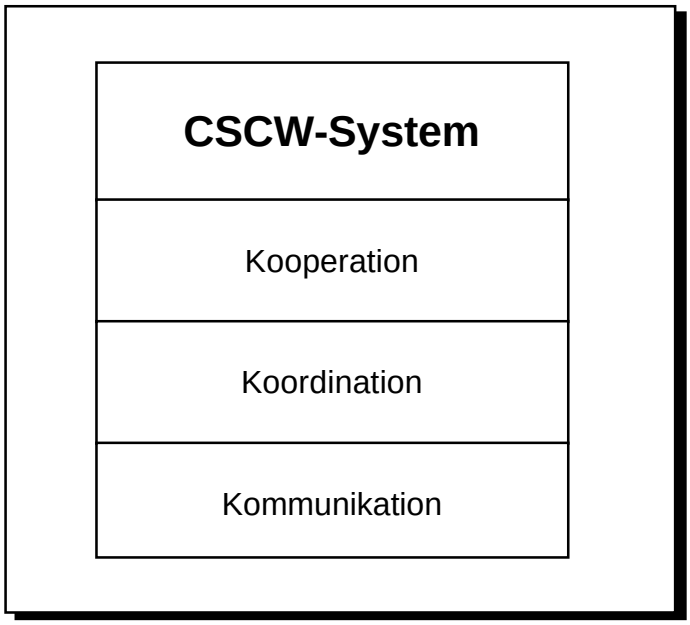

Abb.1: Grundlegender Aufbau eines CSCW-Systems

2.1.1 Die Kommunikation

Die Grundlage jeder Kooperation, und damit jedes CSCW-Systems, stellt die **Kommunikation**, also die Übermittlung, bzw. den Austausch von Informationen, zwischen verschiedenen Objekten dar[3].

[1] Vgl. [DiSa93]
[2] vgl. [Yan95]
[3] vgl. [Lud89]

Die Kommunikation erfolgt mittels eines Kommunikationsmediums, d.h. durch einen Kommunikationskanal, über den Nachrichten ausgetauscht werden. Der Sender einer Nachricht befindet sich am Eingang und der Empfänger der Nachricht am Ausgang des Kanals. Beide, sowohl Sender als auch Empfänger, müssen, damit sie sich verstehen und Informationen austauschen können, dieselbe Syntax und Semantik benutzen, also dieselbe Sprache sprechen. Damit eine Kommunikation stattfinden kann, sollte desweiteren der Kommunikationskanal soweit störungsfrei sein, daß die zu übermittelnde Nachricht noch eindeutig ist. Desweiteren sollte, um die Qualität des Kommunikationskanlas zu sichern, eine Auslieferungsgarantie und eine Duplikatvermeidung bestehen.
Ein weiterer Aspekt der Kommunikation ist die Synchronität, d.h. die Möglichkeit der beteiligten Objekte zu verschiedenen Zeiten zu kommunizieren. Man unterscheidet synchrone und asynchrone Kommunikation.

- Bei der **synchronen Kommunikation** kommunizieren die entsprechenden Partner zur selben Zeit miteinander. Als Beispiel hierfür können das Telefon und Videokonferenzsysteme gesehen werden.

- Bei **asynchroner Kommunikation** kommunizieren die beteiligten Partner zu verschiedenen Zeiten miteinander, z.B. bei elektronischer Post oder u.U. bei einer Wandtafel.

2.1.2 Die Koordination

Jede Kooperation und jede Aktivität in Bezug auf eine Kooperation kann durch Koordination organisiert und dementsprechend effizienter als ohne Koordination gestaltet werden. Unter Koordination wird im folgenden die Planung von Aktivitäten für bestimmte Zeiten und Orte verstanden.
Der Koordinationsbedarf, also der Anlaß zur Koordination, entsteht entweder zu Beginn oder während einer Kooperation und zwar durch die Vorgabe von entsprechenden Zielen oder die Entdeckung eines Konfliktes. Hierbei ist der Konfliktfall besonders hervorzuheben, da es dort zu einer Blockierung eines oder mehrerer beteiligter Partner kommen kann.
Damit z.B. die Mitarbeiter in einem Unternehmen ihre Aufgaben erledigen können, müssen sie entweder alleine oder in Kooperation mit anderen Mitarbeitern an gegebenen Orten entsprechende Teilaktivitäten ihrer Gesamtaufgabe ausführen. Aufgrund der Abhängigkeiten und Beziehungen zwischen diesen Teilaktivitäten, den entsprechenden Orten und Mitarbeitern müssen die einzelnen Teilaktivitäten, um Chaos zu vermeiden, geplant werden. Erst durch diese Planung, die nur anhand von entsprechenden Informationen

über die betroffenen Mitarbeiter durchgeführt werden kann, ist es möglich, daß die Aktivitäten und Arbeitsabläufe effizient und effektiv durchgeführt werden[4].
Besonders ist hierbei die Tatsache noch einaml hervorzuheben, daß die einzelnen Aktivitäten voneinander abhängen, bzw. sich gegenseitig beeinflussen können. Schreibt man z.B. eine Einleitung für ein Dokument sind die folgenden Kapitel semantisch von der Einleitung abhängig. Ändert man die Einleitung, so muß der Rest der Arbeit nachgeführt werden.
Würde das Dokument in einem Team geschrieben werden, so müßten sich alle mitarbeitenden Personen der Abhängigkeiten zwischen ihren Tätigkeiten bewußt sein, damit alle Teile des Dokumentes zusammenpassen.

Definition: Teamkoordination

Der Begriff der Teamkoordination wird in dieser Diplomarbeit wie folgt definiert:
Unter **Teamkoordination** versteht man, daß verschiedene Menschen einer oder unterschiedlicher Gruppen ihre Arbeitsabläufe aufeinander abstimmen, um ein entsprechendes Ziel effizient zu erreichen.

Als Beispiel für Koordinationswerkzeuge können elektronische Terminkalender oder CASE-Tools gesehen werden.

2.1.3 Die Kooperation

Unterschiedliche Gruppen können untereinander konkurrieren oder miteinander kooperieren.

Im Fall der Kooperation oder Kollaboration verfolgen die teilnehmenden Gruppen ein gemeinsames Ziel[5]. Anders gesagt kooperieren die Gruppen dann, wenn sie harmonisch auf ein gemeinsames Ziel gerichtet zusammenarbeiten.
Als Beispiel hierfür ist die Kooperation innerhalb einer Arbeitsgruppe eines Unternehmens anzuführen. Die einzelnen Mitarbeiter der entsprechenden Arbeitsgruppe verfolgen für sich ihre individuell unterschiedlichen Ziele[6], wie z.B. die Befriedigung ihrer persönlichen Sicherheitsbedürfnisse oder das Streben nach Selbstverwirklichung. Im Idealfall kooperieren sie jedoch zusammen, um ein gemeinsames Projekt der Arbeitsgruppe erfolgreich zu beenden.

[4] vgl. [BuSe94], [MaCr90]
[5] vgl. [BuSe94], [MaCr90]
[6] vgl. [Mas54]

Im Fall der Konkurrenz verfolgen die teilnehmenden Gruppen ein gegensätzliches Ziel.

Da die Aktionen der einzelnen Kooperationspartner im Fall einer Zusammenarbeit aufeinander abgestimmt, bzw. koordiniert werden müssen, ist es möglich die Werkzeuge der Kommunikation und der Koordination als Grundlage für die Kooperation zu sehen.
Als Beispiele für Kooperationsanwendungen können Werkzeuge zur gemeinsamen Nutzung von Einbenutzeranwendungen (Application Sharing Tools) oder Werkzeuge, die einen gemeinsamen Arbeitsbereich unterstützen (Shared Workspaces, bzw. Awareness-Systeme) angeführt werden.

2.2 Ein grundlegendes Kooperationsmodell

Da in dieser Diplomarbeit die Kooperation durch Unterstützung der Koordination verbessert werden soll, wird ein Kooperationsmodell erläutert, in dem der Einfluß der Koordination auf die Kooperation genauer geklärt wird.

2.2.1 Aktivitäten, Orte und Mitarbeiter

Im folgenden wird zunächst der Begriff der Aktivität eines Mitarbeiters definiert, um anschließend die Beziehungen zwischen **Aktivitäten**, **Orten** und **Mitarbeitern** näher zu untersuchen und die sich daraus für die Kooperation und das Kooperationsmodell ergebenden wichtigen Informationen abzuleiten.

Aktivitäten eines Mitarbeiters

Unter einzelnen Aktivitäten eines Mitarbeiters versteht man z.B. das Editieren einer Datei, die Kommunikation mit anderen Mitarbeitern oder Projektpartnern, das Lernen von Benötigtem und das Anwenden von Erlerntem, um eine entsprechende Aufgabe lösen zu können. Man kann anhand des Gelernten Rückschlüsse auf die vergangenen Aktivitäten, z.B. über persönliche Erfahrungen und erworbene Fertigkeiten, ziehen. Hat sich ein Mitarbeiter in einem Projekt spezielle Kenntnisse und Erfahrungen in Bezug auf die Programmiersprache Java erworben, so kann man davon ausgehen, daß dieser Mitarbeiter auch in Zukunft von diesen neuen Fertigkeiten profitieren wird, er also in der Lage ist, weitere Aufgaben in Bezug auf diese Programmiersprache zu erledigen.

Nachdem der Begriff einer Aktivität näher erläutert wurde, werden nun die Beziehungen zwischen Aktivitäten, Orten und Mitarbeitern untersucht:

Beziehungen zwischen Aktivitäten und Orten:

Manche Aktivitäten können nur an bestimmten Orten mit speziellen Eigenschaften durchgeführt werden. Man kann z.B. in einem kleinen Büro kein großes Arbeitstreffen mit vielen Beteiligten sinnvoll abhalten, oder man braucht spezielle Ausrüstung, wie einen Overheadprojektor, eine Videokamera oder eine Tafel, um verschiedene Aspekte eines Vortrages zu verdeutlichen. In manchen Fällen kann es sogar sein, daß nur ein Ort im gesamten Gebäude für eine bestimmte Aktivität geeignet ist.

Man braucht also einerseits statische Informationen über die Größe des Raumes und seine Einrichtung. Andererseits benötigt man dynamische Informationen, z.B. in wie weit der Raum im Augenblick belegt, bzw. frei ist.

Beziehungen zwischen Aktivitäten und Mitarbeitern:

Die Beziehungen zwischen Mitarbeitern und ihren Aktivitäten ergeben sich aus den Verpflichtungen der Mitarbeiter, bestimmte Aufgaben mit einer bestimmten Priorität zu erledigen. Die Aufgaben können bereits erledigt sein, im Augenblick bearbeitet werden oder erst für die Zukunft geplant sein. Aus der Priorität von noch zu erfüllenden Aufgaben und der damit verbundenen Verpflichtung kann man die Kooperationsbereitschaft eines Mitarbeiters, z.B. sein Verhalten in Bezug auf Störungen zu einem bestimmten Zeitpunkt ableiten[7].

Es werden also Informationen über den Status der zu erledigenden Aufgaben und deren Priorität für die einzelnen Mitarbeiter benötigt, um Rückschlüsse auf die Kooperationsbereitschaft ziehen zu können. Insbesondere der Start- und Endpunkt aller Aktivitäten des Mitarbeiters müssen bekannt sein, um Störungen zu verhindern. Desweiteren werden Informationen über die Fertigkeiten und Fähigkeiten einzelner Mitarbeiter benötigt, damit bestimmt werden kann, welche Mitarbeiter die Fertigkeit haben einzelne Aufgaben zu erledigen, bzw. neue Fertigkeiten innerhalb eines vorgegebenen Zeitrahmens zu erlernen.

Beziehungen zwischen Mitarbeitern und Orten

Diese Beziehung beschreibt an welchem Ort sich ein Mitarbeiter gerade befindet. Nicht betrachtet wird hier die „virtuelle Anwesenheit“ eines Mitarbeiters an anderen Orten, z.B. über seinen Arbeitsplatzrechner und eine Videokonferenzschaltung. Auf die Ort-zu-Ort-, z.B. Nachbarschaftsbeziehungen, wird hier nicht näher eingegangen, da diese Abhängigkeiten in diesem Modell unbedeutend sind.

[7] siehe Kapitel 5.1 Der substantielle Bereich

Die für die Kooperation wichtigen Informationen sind hier Informationen über den physikalischen Aufenthaltsort eines Mitarbeiters zu einer bestimmten Zeit, damit der Mitarbeiter von einem evtl. Kooperationspartner gefunden werden kann.

Beziehungen zwischen Mitarbeitern

Die Beziehungen zwischen den einzelnen Mitarbeitern ergeben sich aus den bereits oben beschriebenen Beziehungen zwischen Mitarbeitern, Orten und Aktivitäten.

Zusammenfassend kann gesagt werden, daß die folgenden Informationen gebraucht werden, um Kooperation zu unterstützen[8]:

- Fähigkeiten und Fertigkeiten aller Mitarbeiter des Teams,
- Status und Priorität der einzelnen Aufgaben der Mitarbeiter des Teams,
- Start- und Endpunkt aller Aktivitäten und Aufgaben,
- der physikalische Aufenthaltsort jedes Mitarbeiters des Teams zu jedem beliebigen Zeitpunkt und
- statische und dynamische Rauminformationen.

[8] vgl. [FPP95], [SoCh94]

2.2.2 Das Kooperationsmodell

Aufbauend auf den Informationen aus Kapitel 2.2.1 wird nun das in dieser Diplomarbeit verwendete **Kooperationsmodell** erläutert, in dem der Einfluß der Koordination auf die Kooperation genauer geklärt wird:

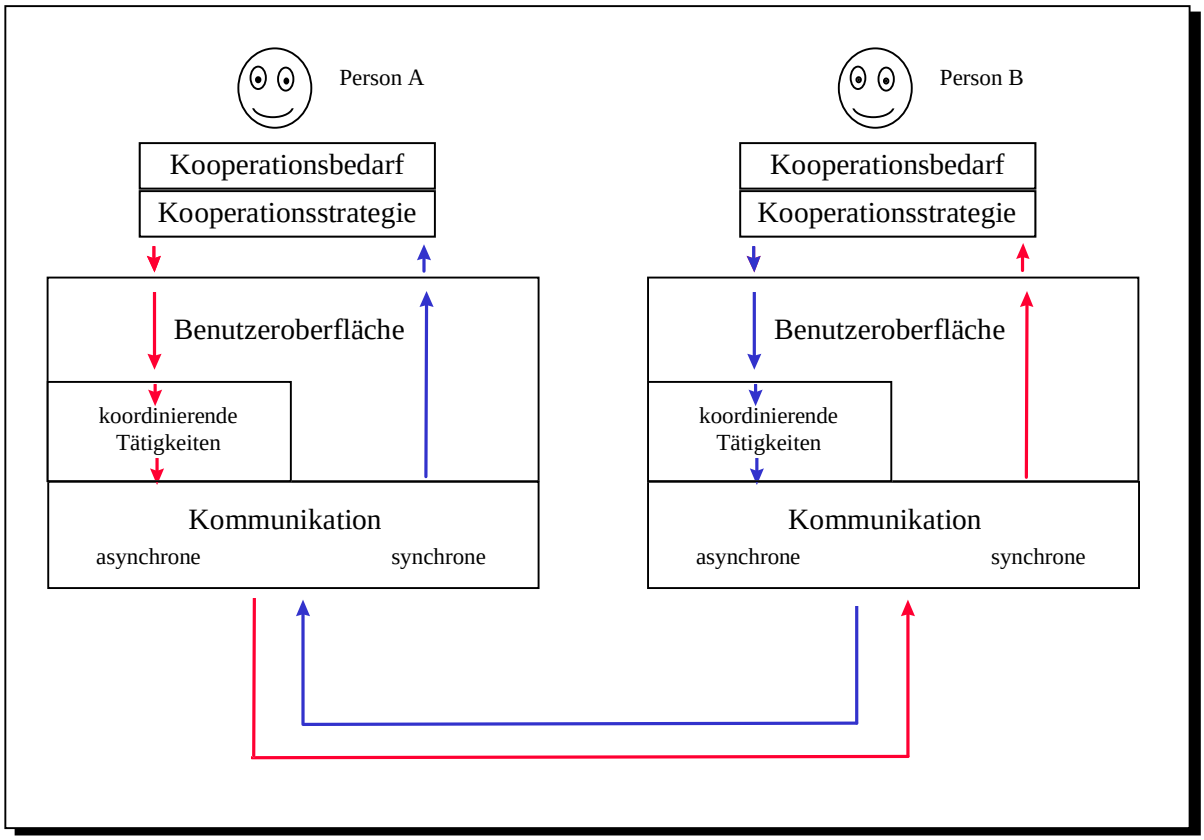

Abb.2: Das Kooperationsmodell

Erläuterungen zum Kooperationsmodell:

Bei einem Mitarbeiter in einem Team kann eine Diskrepanz zwischen seinen Aufgaben und den damit verbundenen Zielen und seinen Fertigkeiten, bzw. seinem Wissen entstehen. Diese Diskrepanz, hier der Kooperationsbedarf, kann sowohl positiv, der Mitarbeiter hat freie Kapazitäten, als auch negativ, der Mitarbeiter braucht Hilfe, sein.

Um seinen Kooperationsbedarf mit anderen Mitarbeitern abzustimmen, definiert sich jeder Mitarbeiter eine **Kooperationsstrategie**, mit der er nach seinen individuellen Bedürfnissen festlegt, inwiefern er anderen hilft, bzw. unter welchen Bedingungen er Hilfe von anderen Mitarbeitern beziehen möchte. Insbesondere muß der Mitarbeiter, wenn er kooperieren will, eine Art von Koordinationsplanung durchführen, in der er die Mitarbeiter seiner

Arbeitsgruppe bestimmt, die die entsprechenden Fähigkeiten, bzw. Rollen, die ihm bei der Lösung der Aufgabe helfen können, besitzen, bzw. innehaben.

Ist der passende Mitarbeiter durch in der Arbeitsgruppe bekannte Arbeitsplatzbeschreibungen[9] oder durch bereits vorhandene persönliche Erfahrungen identifiziert, müssen sich beide über die Zeit und den Ort der Kooperation entscheiden (siehe Beziehungen zwischen Aktivitäten und Orten, Aktivitäten und Mitarbeitern, und Mitarbeitern und Orten). Ausschlaggebend für das Zustandekommen einer Kooperation ist die momentane Kooperationsbereitschaft des erwünschten Mitarbeiters. Diese **koordinierenden Tätigkeiten** werden direkt von der **Benutzerschnittstelle** unterstützt, indem der aktuelle Status[10] aller Anwender dort dargestellt wird.

Nachdem der Mitarbeiter nun den Status des anderen kennt, kann er die beste Kommunikationsform bestimmen.
Ist der passende Mitarbeiter im Augenblick zur Kommunikation bereit, so kann der Anwender **synchron** kommunizieren, d.h. er hat die Möglichkeit eines kurzfristigen, bzw. spontanen Treffens, z.B. ein Anruf oder ein Besuch im Büro.
Ist der passende Mitarbeiter dagegen gerade im Augenblick nicht zur Kommunikation bereit, kann, bzw. sollte der Anwender nur **asynchron** kommunizieren, d.h. er kann das Problem mit dem Mitarbeiter innerhalb eines geplanten Treffens, wie zum Beispiel in der nächsten Gruppensitzung oder per elektronischer Post durchsprechen. Es könnte aber auch sein, daß sich der Kooperationsbedarf, bei Abwesenheit des Kollegen, den man fragen möchte, von selbst erledigt, wenn es sich um eilige Termingeschäfte handelt.
Wenn der Anwender trotz fehlender Kooperations-, bzw. Kommunikationsbreitschaft mit dem passenden Mitarbeiter synchron kommunizieren will, ist es wahrscheinlich, daß der Arbeitsablauf dieses Mitarbeiters durch das Hilfegesuch empfindlich gestört wird. Solch eine Störung des Arbeitsablaufes könnte nur im Fall einer Aufgabe mit größerer Priorität gerechtfertigt sein.

Findet eine Kommunikation statt, wird der angesprochene Mitarbeiter im Rahmen seiner Kooperationsstrategie, seines die Kooperationsanfrage annehmen, bzw. ablehnen.

Man erkennt, daß Beziehungen zwischen einzelnen Mitarbeitern und Aktivitäten, die mehrere Mitarbeitern betreffen, alle Beteiligten direkt beeinflussen. Die daraus resultierende Kommunikation trägt, im Fall der Annahme einer Kooperationsanfrage, dann letztendlich zur gewünschten **Kooperation** bei.

[9] Anmerkung: Arbeitsplatzbeschreibungen wird hier im Sinne von Mitarbeiterbeschreibungen, bzw. Stellenbeschreibungen verstanden.
[10] siehe Kapitel 5.1 Der substantielle Bereich

2.3 Klassifizierung von Koordinations- und Kooperationswerkzeugen

Da in dieser Diplomarbeit ein System zur Unterstützung von Teamkoordination geschaffen werden soll, werden im folgenden nur die Bereiche der Koordinations- und Kooperationswerkzeuge näher daraufhin untersucht, ob bereits Systeme existieren, die die gewünschte Funktionalität bieten.
Der Bereich der Kommunikationswerkzeuge wird nicht näher untersucht, da diese als Grundlage der Koordination und der Kooperation dienen und daher nicht die gewünschte Funktionalität erbringen können.

Um nicht alle Bereiche der Koordinations- und Kooperationswerkzeuge untersuchen zu müssen, werden anhand der Klassifikation dieser Werkzeuge zunächst geeignete Klassen bestimmt, aus denen dann bereits vorhandene Werkzeuge auf die gewünschte Funktionalität hin untersucht werden.

2.3.1 Klassen von Koordinationswerkzeugen

Folgende Klassen von Koordinationswerkzeugen innerhalb des CSCW sollen unterschieden werden[11]:

Workflow Management-Systeme[12]

Bei diesen Systemen wird die Bearbeitung von administrativen Vorgängen strukturiert, indem der Weg durch die verschiedenen Instanzen des Vorgangs koordiniert wird. Der Ablauf eines Vorgangs kann durch das System vorgegeben, von den einzelnen Bearbeitungsschritten abhängig sein oder in einem gewissen Rahmen auf jeder Ablaufebene frei gewählt werden. Das System kann Vorschläge zu den einzelnen Entscheidungen machen oder sogar manche Entscheidungen selbständig treffen.

[11] vgl. [BuSe94]
[12] siehe auch [KRW90], [BUS93] und [GKM93]

CASE-Tools[13]

„**C**omputer-**A**ided-**S**oftware **E**ngineering“-Tools unterstützen, wie der Name schon sagt, den Prozeß der Softwareentwicklung, an dem i.a. mehrere Personen beteiligt sind. Dies geschieht dadurch, daß der Software-Lebenszyklus, also Analyse, Design, Implementierung, Test, Integration und Wartung einer Software, durch Integration geeigneter Entwicklungswerkzeuge, wie z.B. Versionsverwaltungswerkzeuge, vereinfacht oder sogar teilweise automatisiert wird. Durch CASE-Tools werden also die einzelnen Phasen des Lebenszyklus, bzw. die Projektarbeit zur Erstellung eines Softwaresystems koordiniert.

Kooperative Lernsysteme[14]

Kooperative Lernsysteme können insofern als Koordinationswerkzeuge betrachtet werden, als sie Lehrer und Schüler bei der Auswahl, Planung und Durchführung von zielgerichteten Aktionen unterstützen, bzw. koordinieren. Die Systeme können den Lernprozeß entweder direkt steuern, indem sie die Schüler auf einen vorgegeben Weg führen, oder sie können den Lernprozeß auch nur überwachen.

Ferndiagnosesysteme[15]

Diese Systeme unterstützen die Kommunikation und Koordination zwischen lokalen Anwender und entfernten Experten und tragen somit effektiv zur Behebung von Fehlern und der Lösung von Problemen im Hard- und Softwarebereich bei. Der Anwender und der Experte arbeiten dabei gleichzeitig mit vorgegenen Aufgaben, die sich in den verschiedenen Rollen niederschlagen. Dabei kann das Diagnosesystem z.B. dem lokalen Benutzer eines Softwaresystems Hilfestellungen zur Auswahl und zum Ablauf von Diagnoseschritten geben, oder sogar eigenständig kleinere Diagnosen erstellen, die dann dem Experten übermittelt werden können.

Teleshopping-Systeme[16]

Diese können deshalb als Koordinationswerkzeuge verstanden werden, weil sie die Verhandlungen zwischen dem Ein- und dem Verkäufer koordinieren. Ein Verkäufer gibt dem System sein Angebot mit seinen Preisspannen bekannt, und ein Einkäufer, der sich für

[13] siehe auch [SOM92]
[14] siehe auch [MüSc92]
[15] siehe auch [FTP96I], [FTP96II], [FTP96III]
[16] siehe auch [Vil96]

ein entsprechendes Produkt interessiert, verhandelt mit dem Verkaufssystem im Rahmen seiner Preisvorstellungen über die entsprechenden Angebote.

Elektronische Terminplanungssysteme[17]

Sie können zur Verwaltung der Termine einzelner Personen und zur Abstimmung von gemeinsamen Terminen zwischen mehreren Personen dienen. Ein Terminkalender hat meist die Fähigkeit, die Koordination von Terminen dadurch zu unterstützen, daß er die vorhandenen Termininformationen eines oder mehrerer Benutzer, die ein Treffen abhalten wollen analysiert und den für alle Teilnehmer günstigsten Terminvorschlag macht.

Wie aus der Klassifizierung zu erkennen ist, bieten nur elektronische Terminplanungssysteme eine ähnliche Funktionalität, wie sie in der Diplomarbeit gewünscht wird. Aus diesem Grund werden im Kapitel 2.4.1 die Terminplanungssysteme Calendar Manager der Firma SUN, Microsoft's Schedule+, CAP II und Montage näher untersucht.

[17] siehe auch [GrPa95], [CM94I], [Mue93], [TIR94]

2.3.2 Klassen von Kooperationswerkzeugen

Folgende Klassen von Kooperationswerkzeugen können unterschieden werden[18]:

Werkzeuge zur gemeinsamen Nutzung von Einbenutzeranwendungen: [19]

Bei Werkzeugen dieser Klasse, werden die Ausgaben einer Anwendung allen beteiligten Benutzern zur Verfügung gestellt (Application Sharing). Das Werkzeug steuert das Eingabegerät und erlaubt es meistens immer nur einem Benutzer dieses zu verwenden. Das Werkzeug liefert zusätzlich noch die Funktionen zur Team-, bzw. Sitzungsverwaltung, wie z.B. die Weitergabe des Rechtes die Maus zu benutzen an einen anderen teilnehmenden Anwender.

Werkzeuge zur Nutzung von gemeinsamen Arbeitsbereichen[20]

Diese Klasse von Werkzeugen stellt den Benutzern eine Abstraktion beliebiger gemeinsam zugänglicher Objekte zur Verfügung (Shared Workspaces und Awareness-Systeme). Das Werkzeug koordiniert den gemeinsamen Zugriff auf diese Objekte durch mehrere Benutzer und führt, damit die einzelnen Anwender Änderungen nachvollziehen können, eine Versionsverwaltung der einzelnen Objekte durch. Wie auch beim Application Sharing werden Funktionen, wie z.B. Benachrichtigungsmechnismen, zur Team-, bzw. Sitzungsverwaltung angeboten.

Design Datenbanken[21]

Diese Datenbanken erweitern die auf gegenseitig konkurrierende Benutzer ausgelegte Datenbankfunktionalität um Kooperationsmechanismen, so daß z.B. die Anwender auf Zwischenergebnisse einer Abfrage ihrer Arbeitskollegen zugreifen können.

Aus der Klassifizierung ist zu erkennen, daß die Klasse der Werkzeuge zur Nutzung von gemeinsamen Arbeitsbereichen eine Obermenge der zu erstellenden Funktionen bildet. Im Kapitel 2.4.2 werden deshalb die Werkzeuge DIVA und GroupDesk aus dieser Klasse daraufhin untersucht, ob sie die gewünschte Funktionalität schon enthalten.

[18] vgl. [BuSe94]
[19] siehe auch [DeFr92], [Tel95], [Pro96]
[20] siehe auch [SeRo93], [ESM96], [FPP95], [PSF95], [SaTo95], [SoCh94]
[21] siehe auch [NRZ92] und [Chi92]

2.4 Analyse bestehender Koordinations- und Kooperationswerkzeuge

In diesem Kapitel werden bestehende Koordinations- und Kooperationswerkzeuge daraufhin untersucht, ob sie bereits die gewünschte Funktionalität der Diplomarbeit enthalten. Dazu werden zunächst die grundlegenden Anforderungen an die Funktionalität aufgestellt:

- Darstellung der Kooperationsbereitschaft eines Benutzers, bzw. seiner momentanen Unterbrechbarkeit. Dazu müssen die Informationen aus Kapitel 2.2.1 vorliegen, bzw. vom Softwaresystem verwaltet werden.

- Darstellung der Kooperationsbereitschaft der Arbeitskollegen eines Benutzers.

- Möglichkeit zur Integration bereits bestehender Kommunikations-, bzw. Kooperationswerkzeuge, wie z.B. elektronische Mail und Kalendermanagementwerkzeuge.

- Abteilungsweite Verfügbarkeit des Werkzeugs.

2.4.1 Die Funktionalität vorhandener Koordinationswerkzeuge

Die Funktionalität folgender Koordinationswerkzeuge wurde untersucht:

2.4.1.1 Calendar Manager CM[22]

Das Kalendermanagementwerkzeug CM der Firma SUN Inc. läuft unter OPEN WINDOWS und CDE auf SPARCstations. Mit Hilfe des CM kann man Verabredungen treffen und vorher eingetragene Ressourcen aufteilen.
Für jeden Termin kann man das Datum, die Anfangs- und Endzeit, eine Terminbeschreibung, den Benachrichtungsmechnismus, einen beliebigen Wiederholungszeitraum und eine Sichtbarkeitsstufe des Termins einstellen.
Je nach Benachrichtigungsmechnismus kann der CM, aber nur wenn er aktiviert ist, verschiedene Erinnerungsnachrichten, wie ein akustisches oder optisches Signal, oder sogar eine elektronische Post an den jeweiligen Benutzer senden, wenn ein zuvor eingestellter Termin beginnt.

[22] vgl. [GrPa95], [CM94I], [UC1]

Jeder Termin kann als öffentlich, privat oder geheim deklariert werden, und je nach Sichtbarkeitsstufe wird einem anderen Benutzer die Zeit und die Terminbeschreibung, nur die Zeit oder keine Information über den Termin angezeigt.
Man kann, wenn der entsprechende Benutzer seinen Kalender anderen Anwendern freigegeben hat, bei dessen Benutzern nach freien Terminen suchen, oder sogar Termine bei ihnen eintragen. Dazu kann man für jeden Benutzer Zugriffsbeschränkungen, wie darf nur lesen (engl. <browse>), einfügen (engl. <insert>), löschen (engl. <delete>) oder hat keinen Zugriff, definieren.
Mit CM ist es möglich, sich verschiedene Ansichten, z.B. die Wochenansicht auf die eingetragenen Verabredungen des aktuellen Kalender anzeigen zu lassen.
Desweiteren können durch die Integration des MAILTOOLS der Firma SUN Einladungen an andere Benutzer versendet werden, die die Termininformation, per Drag-and-Drop, in ihren Terminkalender eintragen können.

Die Bewertung anhand der aufgeführten Kriterien ergibt, daß die gewünschte Funktionalität durch CM teilweise zur Verfügung gestellt wird:
Die momentane Kooperationsbereitschaft eines Benutzers ist nur indirekt, und nicht direkt, über einen evtl. vorhanden Eintrag im persönlichen Kalender des Anwenders zu erkennen.
Es ist z.B. möglich den Start- und Endpunkt aller Aktivitäten und Aufgaben, oder den physikalischen Aufenthaltsort eines Benutzers, falls er im CM eingetragen ist, abzufragen. Es fehlt die Möglichkeit, die Priorität für eine Aufgabe zu definieren, bzw. statische und dynamische Rauminformationen anzugeben.
Sind keine oder zu wenig Informationen im CM eingetragen, ist der Termin als geheim eingestuft oder reichen die Privilegien des anfragenden Benutzers nicht aus, so kann keine Aussage über die momentane Kooperationsbereitschaft getroffen werden.
Die Integration weiterer Kommunikationswerkzeuge ist, bis auf das bereits vorhandene MAILTOOL, nicht geplant. Dafür ist der CM abteilungsweit verfügbar und wird auch bereits sporadisch verwendet.

2.4.1.2 Microsoft Schedule+[23]

Der elektronische Kalender der Firma Microsoft läuft auf Personal Computern unter den Betriebssystemen Windows 3.11, Windows 95 und Windows NT. Er bietet im Prinzip die gleiche Funktionalität wie CM an. Es ist ohne weiteres möglich, Verabredungen mit anderen Benutzern zu treffen, bzw. in Schedule+ eingetragene Ressourcen, wie z.B. einen Raum, zu verwalten. Erinnerungsnachrichten werden automatisch per Microsoft Mailsystem an die eingetragenen Benutzer versendet, und zwar unabhängig davon, ob der spe-

[23] vgl. [GrPa95]

zielle Benutzer gerade seinen Kalender aktiviert hat. Ebenso wie im CM kann man zwischen diversen Ansichten auswählen. Selbstverständlich können auch neu eingetragene Termine in beliebigen Abständen automatisch wiederholt werden. Es ist auch möglich, Terminkalender anderer Benutzer, je nach Zugriffsberechtigung, auf freie Terminen zu durchsuchen, bzw. falls die entsprechenden Privilegien eingerichtet wurden, wie den eigenen Kalender zu verwalten.
Termininformationen werden per elektronischer Post an die entsprechenden Adressaten gesendet. Benutzen diese Microsoft Mail, wird die Termininformation automatisch erkannt und die Benutzer können direkt per Knopfdruck den Termin eintragen oder ablehnen.
Die Bewertung anhand der aufgeführten Kriterien ergibt, daß die gewünschte Funktionalität, wie beim CM, teilweise durch Schedule+ zur Verfügung gestellt wird :
Die momentane Kooperationsbereitschaft eines Benutzers ist auch hier nur indirekt über einen evtl. vorhandenen Eintrag im persönlichen Kalender des Anwenders zu erkennen. Es fehlen die Möglichkeiten die Priorität einer Aufgabe, oder irgendwelche Rauminformationen anzugeben.
Ist nichts im Kalender eingetragen, oder reichen die Privilegien des anfragenden Benutzers nicht aus, so kann, wie beim CM, keine Aussage über die momentane Kooperationsbereitschaft getroffen werden.
Die Integration weiterer Kommunikationswerkzeuge ist, bis auf die bereits vorhandene Integration der Microsoft Mail, nicht geplant. Die Verfügbarkeit von Schedule+ innerhalb der Abteilung ist gering, da primär auf Workstations, die mit dem Betriebssystem SOLARIS laufen, gearbeitet wird.

2.4.1.3 CAP II[24]

Bei dem Kalendermanagementwerkzeug Calendar Apprentice Version II (CAP II) handelt es sich um einen aus „Erfahrung“ lernenden Terminkalender unter dem Betriebssystem UNIX. Dieser baut auf einem extra dafür, in der Sprache Common LISP, entwickelten Multi-Agentensystem auf. Ziel von CAP II ist es, autonome, lernfähige und verhandelnde Agenten für die Büroautomatisierung bereitzustellen, also einen möglichen Benutzer von der Alltagsarbeit, z.B. Terminverhandlungen, weitgehend zu befreien.
Dazu schlägt das Softwaresystem automatisch beim Eintragen eines neuen Termins, anhand der Termininformation und den internen Regeln, die aus allen vorigen Kalendereinträgen gelernt wurden, die Zeitdauer vor. Diese Termininformation kann jederzeit vom Anwender, falls er die Werte nicht akzeptiert, wieder überschrieben werden.
CAP II ist hierbei ein System kommunizierender, endlicher Automaten, die beim Verhandeln einem Kontraknetzprotokoll folgen, so daß bei einer Einladung, die per elektroni-

[24] vgl. [Mue93], [MCF94]

scher Post an die Adressaten gesendet wird, die entsprechend anderen Benutzeragenten rein reaktiv und lokal entscheiden. „Die Synchronisation erfolgt nur durch Kommunikation und Verhandlung“[25]. Die elektronische Post des Anwenders wird dazu von CAP II automatisch nach Nachrichten im spezifizierten Nachrichtenformat von CAP II gefiltert.

Die Bewertung anhand der aufgeführten Kriterien ergibt, daß die gewünschte Funktionalität durch CAP II etwas umfassender als durch die bereits geschilderten Systeme,zur Verfügung gestellt wird:
Die momentane Kooperationsbereitschaft eines Benutzers ist indirekt über einen evtl. vorhandenen Eintrag im Kalender des Anwenders zu erkennen. Es fehlt die Möglichkeit, die Priorität einer Aufgabe oder irgendwelche Rauminformationen anzugeben.
Ist nichts eingetragen oder reichen die Privilegien des anfragenden Benutzers nicht aus, so kann das System aufgrund bisheriger Erfahrungen eine Aussage über die wahrscheinliche momentane Kooperationsbereitschaft treffen.
Die Integration weiterer Kommunikationswerkzeuge ist, bis auf die bereits vorhandene Unterstützung für elektronische Post, nicht geplant. CAP II ist zum Zeitpunkt dieser Arbeit nicht in der Abteilung verfügbar.

2.4.1.4 Montage[26]

Das von SUN für SPARCstations und SOLARIS entwickelte Koordinationswerkzeug Montage wurde für räumlich verteilte Arbeitsgruppen konzipiert. Das Ziel von Montage ist es, durch die Integration von Video- und Audiofunktionalität, kurzfristige spontane Besuche bei Arbeitskollegen zu ersetzen. Dies ist z.B. mit einer Videokonferenz möglich.
Jeder Benutzer kann durch Angabe seiner momentanen Kooperationsbereitschaft oder seines momentanen Aufenthaltsortes anderen Benutzern von Montage seine Unterbrechbarkeit mitteilen. Ist der Empfänger einer Anfrage augenblicklich unterbrechbar, so wird dieser Benutzer über ein akustisches Signal, entsprechend einem Klopfen an seiner Bürotüre, benachrichtigt. Anschließend kann der Benutzer eine Video-, bzw. Audiokonferenz starten.
Ist der Empfänger einer Anfrage momentan nicht unterbrechbar, so kann ein integrierter Kalender durchsucht, eine kleine Notiz hinterlassen oder eine elektronische Nachricht erzeugt werden.

Die Bewertung anhand der aufgeführten Kriterien ergibt, daß die gewünschte Funktionalität durch Montage in Grundzügen zur Verfügung gestellt wird:

[25] siehe [Mue93]
[26] vgl. [TIR94]

Die momentane Kooperationsbereitschaft eines Benutzers ist direkt über die Benutzeroberfläche erkennbar. Der Benutzer kann die Priorität einer Aufgabe durch seine momentane Unterbrechbarkeit mitteilen. Der physikalische Aufenthaltsort kann, wie auch die Start- und Endzeit eines Termins, im System angegeben werden. Die Möglichkeit der Angabe von statischen und dynamischen Rauminformationen fehlt jedoch.
Die Integration von weiteren Kommunikationswerkzeugen, zusätzlich zu der bereits vorhandenen Unterstützung für Video- und Audiokonferenzing, Email und Kalendermanager, ist in Vorbereitung.
Dieses Werkzeug war zum Zeitpunkt der Diplomarbeit jedoch nicht in der Abteilung verfügbar.

2.4.2 Die Funktionalität vorhandener Kooperationswerkzeuge

Folgende Kooperationswerkzeuge wurden untersucht:

2.4.2.1 DIVA[27]

Diese virtuelle Büroumgebung unterstützt synchrone und asynchrone Kommunikation, Koordination und Kooperation. Dazu orientiert sich DIVA mit seiner graphischen Benutzeroberfläche an einem realen Büro, so daß abstrakte Elemente für Mitarbeiter, Räume, Schreibtische und Dokumente im Softwaresystem existieren, bzw. dargestellt werden können und jeder Benutzer ein eigenes virtuelles Büro erhält.
Es wird eine sogenannte „Teamsichtbarkeit“ (engl. <Group Awareness>) unterstützt. Hierunter wird der Mechanismus verstanden, daß ein Mitglied eines Teams über Aktionen anderer Teammitglieder über spezielle Benachrichtigungsmechnismen informiert wird. Verändert z.B. ein Benutzer ein Dokument, das auf dem virtuellen Schreibtisch eines anderen Benutzer liegt, so werden die Änderungen für den Eigentümer der Datei aufgezeichnet, so daß dieser mittels einer Wiederholungsfunktion später alle Änderungen nachvollziehen kann.
Betritt ein Benutzer ein anderes Büro, so wird vom System sofort ein entsprechender Kommunikationskanal, z.B. eine Videoübertragung, zum entsprechenden lokalen Benutzer eingerichtet, so daß ein quasi „natürlicher“ Start von Aktivitäten gegeben ist. In einem Büro anwesende Personen haben sogar die Möglichkeit, eine gewisse Art von Privatsphäre zu wahren: sie können untereinander flüstern, d.h. ein möglicher Dritter wird aus der Kommunikation ausgeschlossen, oder ein Büro für andere Benutzer abschließen, d.h. ein Dritter kann das Büro nicht betreten. Ist ein Benutzer momentan nicht erreichbar, so kann man ihm eine kleine Notiz an entsprechenden Räumen, Tischen, oder Dokumenten angebracht, hinterlassen.

[27] vgl. [SoCh94]

In jedem virtuellen Büro existiert desweiteren noch ein privater Bereich für die Dokumente des lokalen Benutzers: der Aktenkoffer. Die Dokumente, die in ihm aufbewahrt werden, können nicht von anderen Personen geöffnet werden.

Die Bewertung anhand der aufgeführten Kriterien ergibt, daß die gewünschte Funktionalität durch DIVA teilweise zur Verfügung gestellt wird:
Die momentane Kooperationsbereitschaft und der physikalische Aufenthaltsort eines Benutzers sind indirekt mittels der graphischen Benutzeroberfläche dadurch erkennbar, daß ein Benutzer gerade ein Dokument bearbeitet oder daß er sich gerade in einem Gespräch mit einem anderen Benutzer befindet. Es ist nicht erkennbar, welche Priorität eine Aufgabe hat, oder zu welchem Zeitpunkt sie gestartet, bzw. beendet wird.
Die Integration von weiteren Kommunikations-, bzw. Koordinationswerkzeugen, zusätzlich zu der bereits vorhanden Unterstützung für Video- und Audiokonferenzing, ist vorgesehen.
Dieses Werkzeug war zum Zeitpunkt der Diplomarbeit jedoch nicht in der Abteilung verfügbar.

2.4.2.2 GroupDesk System[28]

Das GroupDesk System wurde zur Unterstützung von verteiltem Arbeiten auf der Grundlage des Prinzips von gemeinsamen Arbeitsbereichen und Benachrichtigungsmechanismen in einer CORBA-Umgebung entwickelt. Der Schwerpunkt des Systems liegt auf der Unterstützung der Teamsichtbarkeit. Dazu werden im GroupDesk System Objekte, Relationen zwischen den Objekten, Anwender und Ereignisse definiert. Die Objekte sind die grundlegenden Informationsträger, wie z.B. ein Dokument. Zu jedem Informationsobjekt existieren entsprechende Relationen, z.B. eine Besitzer-Relation oder eine Interessiert-An-Relation. Wird ein Objekt geändert, so wird ein Ereignis im System ausgelöst und entsprechend den vorhandenen Relationen im System propagiert. Durch diesen Benachrichtigungsmechanismus wird verhindert, daß Benutzer unnötige oder ungewollte Informationen erhalten.
Jeder Arbeitsbereich innerhalb des GroupDesk Systems wird durch einen sog. „GroupDesk“ (engl. <Schreibtisch>) repräsentiert.
Beim Anmelden an das System wird der Benutzer über die Kooperationsbereitschaft aller Mitglieder dieses GroupDesks informiert. Die Kooperationsbereitschaft der einzelnen wird hierbei von dem System anhand der momentanen Aktivität eines Benutzers berechnet. Bearbeitet ein Benutzer gerade ein Dokument, so ist seine Kooperationsbereitschaft niedriger als die eines Benutzers, der gerade kein Dokument bearbeitet. Dies wird durch unterschiedliche Farben des Symbols eines Benutzers dargestellt.
Die Anwender können mittels Videokonferenzing oder elektronischer Post untereinander kommunizieren.

[28] vgl. [PSF95], [FPP95]

Die Bewertung anhand der aufgeführten Kriterien ergibt, daß die gewünschte Funktionalität durch GroupDesk teilweise zur Verfügung gestellt wird:
Die momentane Kooperationsbereitschaft eines Benutzers innerhalb eines GroupDesks ist direkt mittels den farbigen Symbolen auf der graphischen Benutzeroberfläche zu erkennen. Die Priorität einer Aufgabe, statische und dynamische Rauminformationen, der physikalische Aufenthaltsort und der Start- und Endpunkt einer Aktivität sind nicht erkennbar.
Die Integration von weiteren Kommunikations-, bzw. Koordinationswerkzeugen, zusätzlich zu der bereits vorhanden Unterstützung für Video- und Audiokonfernzing, ist vorgesehen. Dieses Werkzeug war zum Zeitpunkt der Diplomarbeit jedoch nicht in der Abteilung verfügbar.

Aus der Analyse der Funktionalität der bereits vorhandenen Systeme ist zu erkennen, daß die Grundzüge der gewünschten Funktionalität des zu erstellenden Softwaresystems bereits in einigen Systemen in der Forschung oder sogar am Markt verfügbar sind. Es existiert jedoch zum momentanen Zeitpunkt kein Werkzeug, das die gesamte Funktionalität beinhaltet. Desweiteren ist keines der Werkzeuge, das die Grundzüge der Funktionalität bietet, am Lehrstuhl vorhanden, so daß ein neues Softwaresystem entworfen werden muß.

3. Agententechnologie

Nachdem in Kapitel 2 vorhandene Werkzeuge aus dem Gebiet des computer gestützten kooperativen Arbeitens daraufhin untersucht wurden, ob bereits vergleichbare Ansätze oder Arbeiten im Bereich der Teamkoordination existieren, wurde in dieser Diplomarbeit, da kein Werkzeug die gewünschte Funktionalität bietet, ein eigenes Softwaresystem implementiert.

Ein Ziel dieser Diplomarbeit ist es, die Anwendung der Agententechnologie am Beispiel der Teamkoordination zu evaluieren. Aus diesem Grund vorhandene mobile Agentensysteme daraufhin untersucht, ob sie als grundlegende Technologie für das zu erstellende Softwaresystem in Frage kommen

Im ersten Teil dieses Kapitels werden dazu zunächst einige Konzepte von Agenten, Agentensystemen und mobilen Agenten vorgestellt. Im zweiten Teil werden dann anschließend vorhandene mobile Agentensysteme analysiert und bewertet.

3.1 Definitionen

Im Bereich der Informatik existieren viele verschiedene Definitionen von **Agenten**, die nach Sichtweise und Einsatzgebiet sehr differieren[29]. Exemplarisch seien hier zwei davon herausgegriffen.
Laut der Definition des Duden[30], der sich auch [Fün95] anschließt, handelt es sich bei einem Agenten um jemanden „... der im Auftrag für einen anderen eine Aufgabe erledigt“.
Die zweite Agentendefinition, die hier vorgestellt wird ist die von [WoJe94]. Für die Autoren ist ein Agent ein unabhängiges Programm, das in der Lage ist, seine Entscheidungen und sein Handeln, basierend auf der Wahrnehmung seiner Umwelt, bei der Verfolgung eines oder mehrerer Ziele selbständig zu kontrollieren.
In [WoJe94] werden folgende Eigenschaften beschrieben, die ein Programm erfüllen muß, damit es als Agent angesehen werden kann:

- **Autonomie**: Die Agenten lösen eine ihnen gestellte Aufgabe selbständig, d.h. ohne weitere Benutzereingriffe und mit eigenständiger Kontrolle über ihr Handeln und ihren internen Status.

- **Kooperationsfähigkeit**: Agenten sind in der Lage mit anderen Agenten und Anwendern zusammenzuarbeiten[31].

[29] vgl. [Sel94], [Ous95], [GeKe94], [Dud89], [WoJe95], [Fon93] und [Fün95]
[30] siehe [Dud89]
[31] siehe [GeKe94]

- **Reaktionsfähigkeit**: Agenten beobachten ihre Umgebung und reagieren auf auftretende Veränderungen.

- **Unternehmungsgeist**: Agenten reagieren nicht nur auf Veränderungen, sie ergreifen sogar die Initiative bei der Verfolgung ihrer Ziele.

Alle Agenten können aber nur innerhalb ihres **Agentensystems** existieren. Dieses Agentensystem sorgt dafür, daß die einzelnen Agenten ausgeführt werden können und bietet den Agenten darüber hinaus die Möglichkeit, auf die Systemresourcen und -dienste kontrolliert zuzugreifen.

Grundsätzlich kann man die Agententechnologie momentan in zwei verschiedene Klassen[32] einteilen:
In der ersten Klasse werden die Agenten aus der Sicht der Verteilten Künstlichen Intelligenz betrachtet. In dieser Klasse werden autonome, persönliche Agenten bzw. Multi-Agentensysteme definiert und verwendet. Der Schwerpunkt liegt auf der „Intelligenz“ der Agenten. Als Intelligenz wird dabei die Fähigkeit der Agenten bezeichnet, aus vergangenen Situationen etwas für ihr Verhalten in der Zukunft zu lernen und dadurch flexibel reagieren zu können. [33]
Die zweite Klasse betrachtet die Agenten aus der Sicht der verteilten Systeme. Hier wird der Schwerpunkt zur Zeit auf die Mobilität der Agenten gelegt, damit verteilte Anwendungen, wie z.B. Trading auf elektronischen Märkten, besser unterstützt werden. Unter Mobilität wird die Fähigkeit der Migration verstanden.

Diese Diplomarbeit baut auf einem Agentensystem auf, das **mobile Agenten** verwendet. Da mobile Agenten eine Unterklasse der Gesamtklasse aller Agenten darstellen, sind die obigen allgemeinen Definition auch für mobile Agenten gültig.
Mobile Agenten besitzen jedoch eine weitere Eigenschaft: sie sind, wie ihr Name schon sagt, mobil, d.h. sie können von einem Ort[34] zu einem anderen migrieren. Im Grunde handelt es sich dabei um eine Prozeßmigration, wie man sie z.B. in verteilten Systemen zur Lastbalancierung verwendet. Darüber hinaus stellt die Migration von Agenten die Grundlage für ein neues Programmierparadigma dar, das für den Einsatz im Bereich der verteilten Systeme besonders geeignet zu sein scheint.
Während eine normale Prozeßmigration auf einer mehr oder weniger eng gekoppelten Mehrprozessormaschine durchgeführt wird, können sich mobile Agenten prinzipiell weltweit zwischen den verschiedenen heterogenen Rechnern bewegen. Die Voraussetzung für die Migration besteht darin, daß auf dem anderen Rechner ein kompatibles Agentensystem vorhanden ist und die beiden Rechner eine Möglichkeit zur Kommunikation unter-

[32] vgl. [Bur96]
[33] vgl. [Mue93], [MCF94], [Mae94], [Mae95I], [Mae95II], [PYF92], [Rie94], [ICM91]
[34] Anmerkung: Die Orte in einem Agentensystem werden auch als Locationen bezeichnet.

einander haben. Die Rechner könnten z.B. über TCP/IP, sprich über das Internet, verbunden sein.
Anstatt, wie bei den bisher in weit verteilten Systemen verwendeten „Remote Procedure Calls“ (RPC), die Daten oder die einzelnen Anfragen direkt über das Netz zu senden, wird bei Agentensystemen mit mobilen Agenten der Programmcode zur Berechnung des Ergebnisses über das Netz gesendet und lokal auf dem dortigen Diensterbringer ausgeführt.
Mit diesem Ansatz läßt sich zunächst keine vollkommen neue Funktionalität modellieren. Durch die lokale Ausführung des Programmcodes auf dem Server und der damit möglicherweise verbundenen Einsparung von Datentransporten über das Netzwerk vereinfacht sich allerdings die Programmierung vieler Aufgaben erheblich, da z.B. auf viel weniger Fehlerbedingungen Rücksicht genommen werden muß[35].

Im folgenden wurden die Vor- und Nachteile eines Agentensystems mit mobilen Agenten näher untersucht[36]:

Vorteile eines Agentensystems mit mobilen Agenten:

- Mobile Agenten unterstützen „mobile computing“ dadurch, daß sie schmale Bandbreiten, geringe Rechen- und Speicherleistung der mobilen Endgeräte und eine geringe Netzpräsenz der Endgeräte unterstützen.

- Die Kommunikation zwischen den Partnern, z.B. einem Agenten und einem Benutzer, kann asynchron stattfinden. Der Vorteil hierbei liegt darin, daß der Initiator einer Aktion nicht auf deren Ende und die Ergebnisse warten muß. Mobile Agenten können also, wenn sie entsprechend programmiert sind, ohne dauernden Kontakt zum Anwender arbeiten.

- Die Kommunikation kann lokal und nicht nur entfernt stattfinden. Hierdurch ist es möglich, effektiv Netzbandbreite zu sparen. In verschiedenen Situationen kann z.B. die Kommunikation mittels eines mobilen Agenten schneller ablaufen als Kommunikation mit RPC´s. Sind in einem Netzwerksystem mit geringer Bandbreite z.B. mehrere Anfragen an einen Server zu stellen, so könnte es schneller sein, einen Agenten zu diesem Server hin migrieren, ihn dort lokal die Anfragen, bzw. Aufgaben ausführen und, wenn das Ergebnis vorliegt, ihn wieder zurück migrieren zu lassen, als mehrere Anfragen über das Netzwerk zu senden.

[35] vgl. [Hoh95], [Hoh96]
[36] vgl. [Hoh95], [Hoh96], [HCK95]

- Mobile Agenten unterstützen die Echtzeitdatenverarbeitung bei einem Server, indem sie zu diesem Server hin migrieren un dort lokal kommunizieren. Dadurch sind die Übertragungszeiten der Daten kürzer; insofern ist die Verarbeitung schneller.

- Client-Server-Computing ist auch mittels mobiler Agenten möglich.

Nachteile eines Agentensystems mit mobilen Agenten:

- Verschiedene Sicherheitsaspekte müssen unbedingt berücksichtigt werden. Ein fremder Agent muß sich beispielsweise beim Agentensystem authentifizieren; seine Funktionen müssen, z.B. im Rahmen eines Viruschecks, überprüft werden.

- Der Schutz vor „böswillig“ programmierten Locationen des Agentensystems kann nicht gewährleistet werden. Eine Location könnte z.B. einen Agenten samt seines elektronischem Geldes löschen, oder das Geld sogar entwenden. Es wäre sogar möglich, daß eine Location den Code eines Agenten manipuliert, so daß sein Verhalten, nach dem Besuch der „böswillig“ programmierten Location, unvorhersehbar wird.

- Die Funktionalität der verschiedenen Agenten läßt sich nur schwer abschätzen, d.h. eine effektive Überprüfung der Funktionalität von Agenten ist nicht möglich. Hieraus resultiert ebenfalls ein Sicherheitsproblem.

- In einem offenen Agentensystem gibt es keine zentrale Autorität, die überprüft, ob verschiedene Regeln, die im Agentensystem gelten eingehalten werden.

Nachdem die Vor- und Nachteile eines Agentensystems mit mobilen Agenten erläutert wurden, wird nun zum besseren Verständnis der Agententechnologie kurz die Sicht eines Agenten auf ein einfaches Agentensystem mit mobilen Agenten näher näher eingegangen:

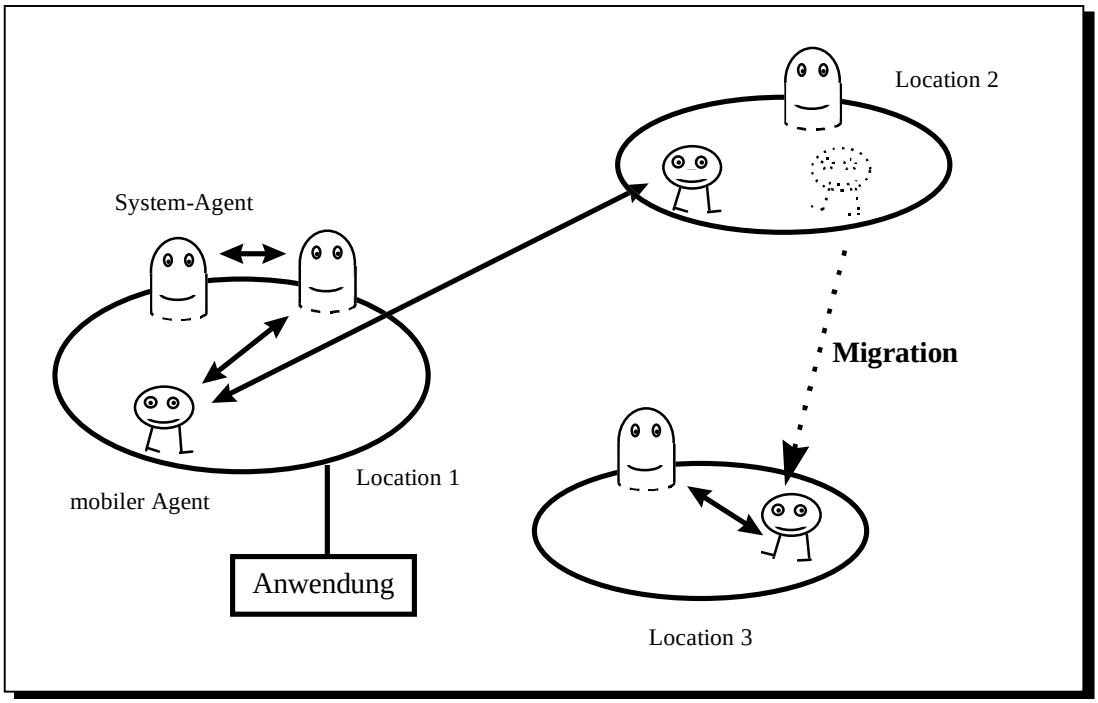

Abb.3: Sicht eines Agenten auf ein einfaches Agentensystem

Grundsätzlich sollte ein Agentensystem aus den einzelnen Agenten und aus den Orten, an denen sich die verschiedensten Agenten aufhalten, bzw. an denen sie ausgeführt werden können bestehen. Die Orte, bzw. die sogenannten Locationen, sind i.a. auf verschiedenen heterogenen Rechnern lauffähig, damit weitverteiltes Arbeiten unterstützt, bzw. möglich wird.

Die verschiedensten Agenten haben einerseits die Möglichkeit sich innerhalb eines Ortes zu treffen, miteinander zu kommunizieren und allein oder gemeinsam Aufgaben zu lösen.

Um die Kommunikationskosten im Netzwerk zu minimieren können sich die Agenten an anderen Orten treffen, um dort lokal Nachrichten auszutauschen. Damit unnötige Migrationen vermieden werden, sind die Agenten auch in der Lage über die Locationsgrenzen hinweg miteinander zu kommunizieren. Durch die Migration erhalten die mobilen Agenten außerdem die Möglichkeit auf spezielle Systemdienste, die nur an einem bestimmten Ort angeboten werden, lokal zuzugreifen.

Die Agenten sollten aus sicherheitstechnischen Gründen in zwei Klassen aufgeteilt werden: die besonders privilegierten Systemagenten, die ohne die Möglichkeit der Migration fest an einen Ort gebunden sind und die direkten Zugriff auf die Systemdienste und -ressourcen haben, und die mobilen Agenten, die sich von Ort zu Ort bewegen können und i.a. keinen direkten Zugriff auf die Systemressourcen besitzen. Sie können nur über

die Systemagenten auf das umgebende Softwaresystem, bzw. auf Anwendungen, die außerhalb des Agentensystems liegen zugreifen. Durch diesen Mechanismus wird verhindert, daß ein Programmierer einen Agenten schreibt, der zu beliebigen Orten migrieren, und sich dort beliebige Systemresourcen reservieren, oder unauthorisiert entfernte Anwendungen ausführen kann.

3.2 Agentensysteme mit mobilen Agenten

Als mobile Agentensysteme werden Java-To-Go[37], CyberAgent[38], Ara[39], Telescript[40] und Mole[41] vorgestellt und anhand der vorhandenen Literatur auf die Eignung als zugrundeliegende Technologie der zu erstellenden Software untersucht. Diese Agentensysteme sind alle aus der Klasse der verteilten Systeme heraus entstanden, d. h. sie unterstützen alle die Migration von Agenten.
Die folgenden Kriterien wurden bei der Bewertung der einzelnen Ansätze verwendet:

- **Verfügbarkeit:** Das Agentensystem muß bei Beginn der Diplomarbeit im Lehrstuhl verfügbar sein, oder es muß ohne weitere Kosten angeschafft werden können.

- **Möglichkeit der Integration von anderen Anwendungen:** Da in dieser Diplomarbeit auch Anwendungen außerhalb des Agentensystems verwendet werden sollen, muß im Agentensystem die Möglichkeit bestehen solche Anwendungen zu integrieren.

- **Unterstützung von Java:** Das Agentensystem muß laut den Anforderungen an die Diplomarbeit die Sprache Java von Sun Microsystems unterstützen. Insbesondere muß die Möglichkeit der Kommunikation mit Java-Applets bestehen, denn das Agentensystem muß die Möglichkeit haben, mittels Applets Informationen über das umgebende System zu beziehen und Informationen an diese zu senden.

3.2.1 Java-To-Go

Java-To-Go ist ein Projekt an der Berkeley Universität in Kalifornien USA, das seit Juli 1996 öffentlich zugänglich ist. Es handelt sich hierbei um eine experimentelle Arbeitsumgebung für die Entwicklung und das Testen von mobilen Agenten und für agentenbasierte, verteilte Anwendungen. Der Schwerpunkt wird hierbei auf eine leicht zu konfigurierende Arbeitsumgebung, die mobile Agenten verwendet, gelegt.

[37] vgl. [LiMe96]
[38] vgl. [FTP96I], [FTP96II], [FTP96III]
[39] vgl. [Pei95]
[40] vgl. [Whi94I], [Whi94II] und [Li96]
[41] vgl. [Hoh95], [Hoh96]

In einer verteilten Anwendung versuchen z.B. ein oder mehrere mobile Agenten parallel eine Reihe von verteilten Rechnern zu kontaktieren, um somit eine gewünschte Information zu erhalten, oder um gemeinsam eine Aufgabe zu lösen. Das Ergebnis wird am Ende dem lokalen Agentensystem mitgeteilt.
Die Infrastruktur von Java-To-Go gliedert sich in zwei Teile: mobile Agenten und Locationen, die hier „stationary Hall servers“ genannt werden.
Den Agenten wird die Möglichkeit gegeben, sich frei zu bewegen und eigene Berechnungen, bzw. Abschätzungen über den ökonomischen Sinn einer Migration zu einen oder mehreren Servern zu treffen.
Ein Agent hat in Java-To-Go die Möglichkeit, weitere Agenten zu erzeugen, die die Ausführungszeit durch paralleles Arbeiten entsprechend beschleunigen können.
Java-To-Go ist in der Programmiersprache Java programmiert worden, da in dieser Sprache eine nahtlose Unterstützung für Netzwerkverbindungen und Sicherheitsdienste integriert wurden. Desweiteren ermöglicht die virtuelle Maschine von Java, das Java-To-Go auf allen Plattformen, die Java unterstützen funktioniert. Zusätzlich erhalten die Agenten ihre Befehle und Daten direkt von Java-Applets, an die sie auch entsprechende Daten wieder zurück senden können.

Bewertung anhand der aufgeführten Kriterien:
Das Agentensystem Java-To-Go unterstützt die Programmiersprache Java und hat auch die Möglichkeit, direkt über Java-Applets zu kommunizieren. Andere Anwendungen können über entsprechende Schnittstellen mittels Hall Server in das Agentensystem eingebunden werden. Dieses Agentensystem war erst Ende Juli 1996 in der Version Pre-alpha 0.95 verfügbar, so daß das Kriterium der Verfügbarkeit nicht erfüllt und das Agentensystem nicht verwendet werden konnte.

3.2.2 CyberAgent

Von der Firma FTP Software Inc. wurde im Laufe des Jahres 1996 die CyberAgent Technologie entwickelt. Hierbei handelt es sich um intelligente, mobile Agenten, die von Netzwerkadministratoren zur Verwaltung des Netzwerks und zur Lösung von Problemen eingesetzt werden können.
Das Agentensystem wurde in der Programmiersprache Java implementiert, da diese eine gewisse Plattformunabhängigkeit des Agentensystems ermöglicht. Es besteht im wesentlichen aus zwei Teilen, den mobilen CyberAgents und den CyberAgent Listeners, die die Locationen im Agentensystem darstellen.
Ein CyberAgent wird als ein autonomes Programm gesehen, das sich über das Netz von Rechner zu Rechner bewegt und dort Aufgaben erledigt. Die Agenten können alleine oder in Gruppen arbeiten. Sie haben die Möglichkeit mit Applikationen, wie z.B. Microsoft Excel zu kommunizieren, um Daten zugänglich zu machen oder sogar Applikationen mit sich

über das Netzwerk, das TCP/IP kompatibel sein muß, zu transportieren. Desweiteren werden drei verschiedene Sicherheitsstufen im Agentensystem angeboten, um den Sicherheitsbedürfnissen der Anwender zu genügen.
Die CyberAgenten unterscheiden sich anhand von vier Schlüsselattributen, die für jeden Agenten definiert werden:

- **Migration:** Je nach Migrationsfähigkeit werden die Agenten in zwei verschiedene Klassen aufgeteilt. Entweder besitzen die Agenten die Möglichkeit, zu entfernten Rechner zu migrieren und dort Applikationen zu starten, oder sie sind an einen einzelnen Rechner gebunden. Haben die Agenten die Fähigkeit der Migration, so kann der Benutzer eine Art Reiseplan festlegen, indem er spezifiziert, ob die Zielrechner seriell nacheinander besucht werden und der Benutzer am Ende ein entsprechendes Ergebnis erhält, oder ob eine Gruppe von Agenten parallel die Zielrechner besucht und dem Benutzer nach jedem Besuch ein Zwischenergebnis liefert.

- **Datentransport:** Die Agenten unterscheiden sich in der Art des Datentransportes. Der Anwender kann spezifizieren, ob der Agent die erhaltene Information bis zum Ende seines Auftrages bei sich trägt oder ob er seine Information sofort, mittels eines Nachrichtenagenten, zurück zum Benutzer sendet. Ein Agent hat sogar die Möglichkeit eine Applikation, als eine seiner Funktionen z.B. ein Antivirusprogramm, mit zum Zielrechner zu transportieren.

- **Entscheidungsfindung:** Jeder Agent hat die Möglichkeit eigene Entscheidungen über seinen weiteren Weg bei einer Migrationsfolge zu treffen, wenn der Zielrechner zur Zeit nicht verfügbar ist. Er kann z.B. versuchen den Zielrechner zu einem späteren Zeitpunkt zu kontaktieren, oder er kann am Ende seines Auftrages dem Auftraggeber eine Liste mit den nicht erreichten Zielrechnern übermitteln.

- **Netzwerkzustandsberichte:** Am Ende ihres Auftrags senden die Agenten je nach erteiltem Auftrag ihre Ergebnisse zurück zum Anwender. Hierbei unterscheiden sich die Agenten in der Form der Ergebnispräsentation.

Bewertung anhand der aufgeführten Kriterien:
Die CyberAgents, die seit Ende Juni 1996 verfügbar sind, wurden in der Programmiersprache Java programmiert und bieten einerseits die Möglichkeit, Applikationen, die außerhalb des Agentensystems liegen, direkt in das Agentensystem zu integrieren. Andererseits wird die Programmiersprache Java soweit unterstützt, bzw. um entsprechende Klassen ergänzt, daß auch die Möglichkeit der Kommunikation der CyberAgents mit Java-Applets besteht. Dieses Agentensystem könnte prinzipiell für diese Diplomarbeit verwendet werden, allerdings handelt es sich hierbei um ein kommerzielles Produkt - dies hätte nicht unerhebliche Anschaffungskosten zur Folge.

3.2.3 Ara

Ara (Agent for Remote Actions) ist ein Projekt, das seit August 1994 an der Universität Kaiserslautern läuft. Es hat die Entwicklung einer portablen und sicheren Infrastruktur für die Ausführung mobiler Agenten in heterogenen Netzen zum Ziel.
Mobile Agenten werden in diesem Projekt als Programme gesehen, die die Fähigkeit besitzen, während der Ausführung ihren Wirtsrechner zu verlassen. Dies erlaubt ihnen, ihre Kommunikation, die vorher entfernt durchgeführt werden mußte, lokal abzuhalten.
Bei Ara liegt der Anwendungsschwerpunkt auf den Diensten für Mobilcomputer mit drahtloser Netzanbindung, einer Umgebung mit inhärenten Einschränkungen im Hinblick auf die Ressourcen und die Verbindungsgüte im Netzwerk.
Die Architektur des Ara-Systems ist in einen Systemkern und mehrere Prozesse gegliedert. Agenten werden als Prozesse ausgeführt. Das erleichtert sowohl ihre unabhängige Ausführung als auch ihren gegenseitigen Schutz. Alle Agenten können über einen RPC-Mechanismus, der über den Systemkern läuft, kommunizieren.
Das gesamte Ara-System, einschließlich aller Agenten und des Kerns läuft dabei als ein einziger Anwendungsprozeß auf dem unmodifiziertem Wirtsbetriebssystem. Während ortsfeste Systemagenten kompiliert werden, laufen mobile Agenten aus Portabilitäts- und Sicherheitsaspekten innerhalb eines Interpreters für die jeweilige Programmiersprache ab.

Dies wird in folgender Abbildung veranschaulicht:

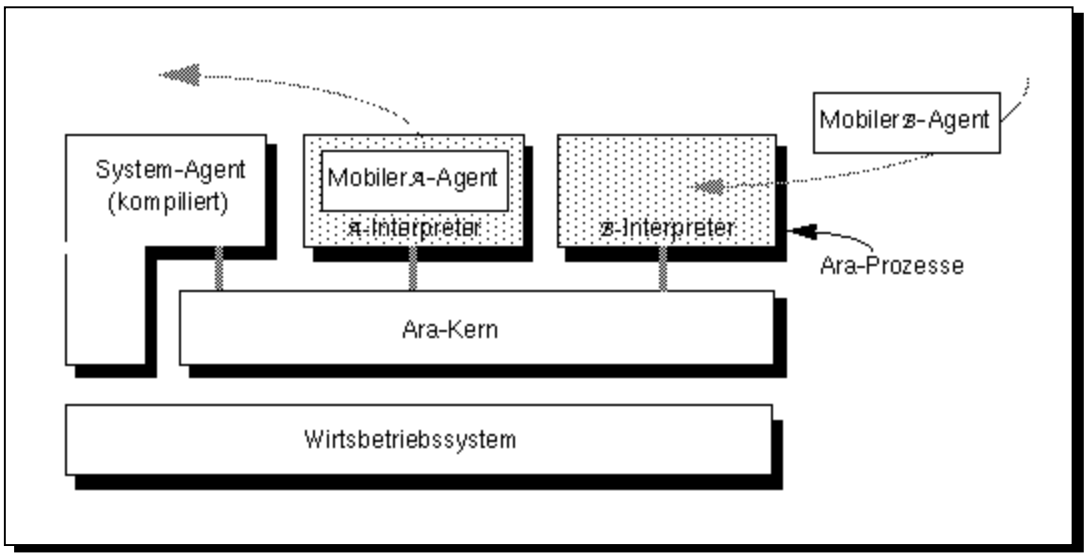

Abb.4: die Architektur des Ara-Systems[42]

Agenten, die als ortsfeste Systemagenten laufen, können in der Programmiersprache C oder C++ implementiert werden. Mobile Agenten, die interpretiert werden müssen, können

[42] siehe [Pei96]

in einer TCL-Variante oder in MACE, einem Bytecode der aus dem Compiler-Zwischencode RTL generiert wird, realisiert werden.
Bewertung anhand der aufgeführten Kriterien:
Da dieses Agentensystem weder Java noch die Möglichkeit der Kommunikation mit Java-Applets unterstützt, wird es nicht als dieser Diplomarbeit zugrunde liegendes Agentensystem verwendet.

3.2.4 Telescript

Das sicherlich bekannteste Beispiel für mobile Agenten ist die Programmiersprache Telescript, die zusammen mit dem umgebenden Betriebssystem Magic Cap, seit 1994 von der Firma General Magic entwickelt wird.
Die Hauptanwendungsbereiche des Betriebssystems Magic Cap liegen vor allem in der Unterstützung kleiner, portabler Rechner für den elektronischen Markt.
Innerhalb von Telescript werden Agenten, Orte, Engines, Migration, Treffen verschiedener Agenten, Kommunikationsverbindungen zwischen Rechnern, Vollmachten und Genehmigungen eines Agenten unterstützt.
Die Agenten von Telescript, die als bewegliche Diensterbringer im Agentensystem gesehen werden, können innerhalb eines Ortes mit speziellen Vollmachten ausgestattet sein, die es ihnen erlauben, auf das umgebende Softwaresystem zuzugreifen oder sogar anderen Agenten die Genehmigung für einen Zugriff zu erteilen. Ein Agent, der einen Auftrag zu erfüllen hat, migriert selbständig zu einem oder mehreren Zielorten, um dort mit den privilegierten Agenten ein Treffen abzuhalten, bei dem Informationen ausgetauscht werden können.
Telescript unterstützt die Programmiersprachen C und C++, wobei die Agenten und die Benutzeroberfläche in der Programmiersprache Telescript implementiert werden müssen.

Bewertung anhand der aufgeführten Kriterien:
Durch die Unterstützung von C, bzw. C++ können auch Anwendungen die außerhalb des Agentensystems liegen in das Agentensystem integriert werden. Leider wird im Augenblick weder die Programmiersprache Java noch die Kommunikation mit Java-Applets unterstützt. Außerdem ist Telescript ein kommerzielles Produkt, das nicht frei am Markt verfügbar ist, so daß Telescript in dieser Diplomarbeit keine Anwendung findet.

3.2.5 Mole

Das aktuelle Mole-Agentensystem, Version Alpha 1.0, wurde im Herbst 1995 an der Universität Stuttgart in der Abteilung Verteilte Systeme am Institut für Parallele und Verteilte Höchstleistungsrechner im Rahmen einer Diplomarbeit von [Hoh95] entwickelt. Es handelt sich hierbei um ein offenes Agentensystem, das für mobile Agenten konzipiert wurde.
Das Agentensystem und die Agenten wurden in der Programmiersprache Java implementiert, da diese entsprechende Sicherheitsmechanismen zur Verfügung stellt. Es besteht im wesentlichen aus zwei Teilen, den Agenten und den Orten, auf denen die Agenten ausgeführt werden können.
Im Molesystem existieren zwei Typen von Agenten, die ortsfesten Systemagenten, die eine Schnittstelle zum umgebenen Softwaresystem darstellen und mobile Agenten, die von Ort zu Ort migrieren können.
Eine Besonderheit von Mole ist, daß die Orte in Mole in Location und Engine unterteilt werden. Die Mole-Engine der aktuellen Implementierung kann eine oder mehrere Locationen instanziieren und läuft innerhalb eines normalen Betriebssystemprozesses. Der Vorteil dieser Unterteilung ist, daß nur ein Java-Interpreter pro Engine benötigt wird. Dies bringt wesentliche Performacegewinne mit sich.
An jeder Location können sich ortsfeste Systemagenten befinden, über die die mobilen Agenten auf die Systemressourcen der Umgebung zugreifen können. Die Agenten haben die Möglichkeit untereinander über Nachrichten, sowohl innerhalb einer Location als auch locationsübergreifend, zu kommunizieren.

Bewertung anhand der aufgeführten Kriterien:
Das Mole-Agentensystem war bereits seit Beginn der Diplomarbeit am Lehrstuhl im Einsatz. Durch seine Implementierung in Java unterstützt es sämtliche Features von Java, so daß die Möglichkeit der Kommunikation über Applets im Agentensystem generell gegeben ist. Im Augenblick ist jedoch nur die Kommunikation von einem Java-Applet zu einem Agenten realisiert. Der umgekehrte Weg, die Weitergabe von Informationen an ein Applet, ist zwar möglich, aber noch nicht offiziell freigegeben. Andere Anwendungen können über entsprechende Schnittstellen eines Systemagenten ins Agentensystem integriert werden.

Da das Mole-Agentensystem das einzige Agentensystem ist, das die aufgeführten Kriterien erfüllt und da es bereits in der Abteilung IPVR eingesetzt wird, wurde es in dieser Diplomarbeit als Agentensystem mit mobilen Agenten ausgewählt.

4. Produktspezifikation

Um die Komplexität des Softwaresystems näher zu untersuchen und zu verstehen, wird eine Spezifikationstechnik aus [ChaMon95] verwendet. Diese Technik basiert auf dem Referenzmodell für offenes, verteiltes Arbeiten [IsoIecI92][IsoIecII92][IsoIecIII92] und ist ein Standardvorgehen für die Spezifikation von generischen, verteilten Systemen.
Ein zu spezifizierendes Softwaresystem wird dabei von fünf Standpunkten aus untersucht:

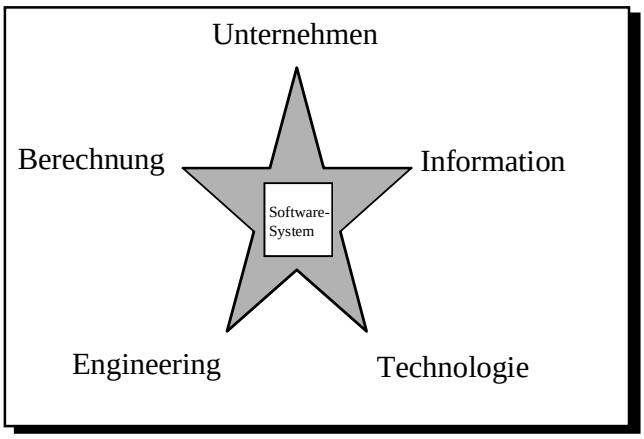

Abb.5: verschiedene Untersuchungsstandpunkte

Die fünf Standpunkte werden wie folgt unterschieden:

- Der **Unternehmensstandpunkt** beschreibt den Zweck, den Einsatzbereich, das zu modellierende Verhalten und die Strategien, die hinter dem zu erstellenden Softwaresystem stehen.

- Im **Informationsstandpunkt** wird die Semantik der Information und die Informationsverarbeitung des zu erstellenden Softwaresystems untersucht. Hier wird versucht das zu erstellende Softwaresystem in eine Menge von interagierenden Objekten aufzuteilen und das Informationsverhalten dieser Objekte zu beschreiben.

- Im **Berechnungsstandpunkt** werden die Schnittstellen zwischen den Informationsobjekten näher untersucht und definiert.

- Beim **Engineeringstandpunkt** wird der Schwerpunkt auf die Infrastruktur des zu erstellenden Softwaresystems gelegt.

- Der **Technologiestandpunkt** gibt die verwendete Programmierumgebung, in die das zu erstellende Softwaresystem integriert werden soll, wieder.

4.1. Der Unternehmensstandpunkt

Das Konzept des Unternehmensstandpunktes bildet einen möglichen Rahmen, um eine Unternehmensspezifikation oder ein Unternehmensmodell hervorzubringen. Ein Unternehmensmodell beschreibt ein Softwaresystem aus der Sicht der Organisation und der Mitarbeiter, die mit der Software arbeiten werden.

Das Konzept des Unternehmensstandpunktes beinhaltet bei dem zu erstellenden Softwaresystem die folgenden Punkte:

- Die **Anwender** sind die Mitarbeiter der Abteilung IPVR der Fakultät Informatik der Universität Stuttgart, da diese in Büros arbeiten, die eine gewisse räumliche Distanz, die nicht zu vernachlässigen ist, zueinander aufweisen und somit das Arbeiten bzw. das Zusammentreffen der Mitarbeiter untereinander zu Besprechungen erheblich erschwert.

- Folgendes **Verhalten** der Mitarbeiter soll durch das zu erstellende Softwaresystem teilweise übernommen, bzw. erleichtert werden: Einige Mitarbeiter müssen sich, da sie in einer Arbeitsgruppe zusammenarbeiten und Informationen austauschen wollen, des öfteren am Tag zu einer spontanen oder zu einer seit längerem geplanten Sitzung treffen.
  Bei den spontanen Treffen versuchen die Mitarbeiter entweder entsprechende Arbeitskollegen telefonisch zu erreichen, oder sie gehen von Büro zu Büro, um entsprechende Sitzungstermine zu verabreden oder spontan etwas mit diesem Mitarbeiter zu besprechen. Dabei ergibt sich immer wieder die Situation, daß der entsprechende Mitarbeiter nicht telefonisch zu erreichen ist, oder daß sich der Mitarbeiter nicht in seinem Büro befindet und seinen aktuellen Aufenthaltsort über einen kleinen Vermerk an der Türe, einen sogenannten Türzettel, bekannt gibt. Aus diesem Verhalten resultieren außerdem permanente Störungen des Arbeitsablaufs der Mitarbeiter, da die momentane Unterbrechbarkeit, bzw. Kooperationsbereitschaft der entsprechenden Mitarbeiter nicht direkt erkennbar ist.
  Die geplanten Treffen der Mitarbeiter werden bei einem spontanen Treffen der jeweiligen Personen, oder bei der einmal pro Woche stattfindenden Gruppensitzung, vereinbart. Hier besteht die Schwierigkeit darin, daß nicht immer alle Mitglieder der Arbeitsgruppe an der Gruppensitzung teilnehmen können, und daß diese Mitglieder über die Vorhaben des Rests der Gruppe informiert werden müssen, bzw. daß auch ihre Termine in die Gruppenplanung mit einbezogen werden müssen, diese aber häufig nicht bekannt sind.

- Daraus läßt sich folgende grundlegende **Anforderung** an das zu erstellende Softwaresystem ableiten:
  Das Softwaresystem muß sowohl die spontanen, als auch die geplanten Treffen der Mitarbeiter derart unterstützen, daß einerseits die Störungen der Arbeitsabläufe minimiert und andererseits Treffen der Mitarbeiter möglichst effizient geplant werden können. Dazu soll für die spontanen Treffen der Mitarbeiter die Funktionalität der Türzettel vom zu erstellenden Softwaresystem nachgebildet werden, d.h. daß die Mitarbeiter und Angestellten die Möglichkeit haben müssen, beim Verlassen ihres Büros einen elektronischen Türzettel im System anzubringen. Dieser muß von den anderen Mitarbeitern von ihrem jeweiligen Arbeitsplatzrechner aus abgefragt werden können, damit sich die Benutzer des Systems den Weg zu, oder den Anruf in dem nicht besetzten Büro ersparen können. Andererseits müssen auch die geplanten Treffen der Mitarbeiter unterstützt werden, d.h. es muß ein Kalendermanagementwerkzeug in das zu erstellende Softwaresystem integriert werden, über das sich Termine verabreden lassen und in das jeder Mitarbeiter seine aktuellen Termine einträgt. Dabei sollten aus Gründen des Datenschutzes verschiedene Vertraulichkeitsstufen innerhalb des Kalendermanagementwerkzeugs beachtet werden. Um die geplanten Treffen weiter zu unterstützen benachrichtigt das Softwaresystem den Benutzer, falls dieser automatisch generierte elektronische Post mit Termininformation erhalten hat.

- Die **Verpflichtungen** der Mitarbeiter und Angestellten bestehen darin, daß sie ihre Termine in das Kalendermanagementwerkzeug eintragen und daß sie das zu erstellende Softwaresystem mit den elektronischen Türzetteln benutzen. Daraus könnten entsprechende Akzeptanzprobleme resultieren**.** Die Mitarbeiter könnten sich weigern ihre Termine zusätzlich zu ihrer eigenen Terminplanung in das Kalendermanagementwerkzeug einzutragen. Wird aber entsprechende Überzeugungsarbeit geleistet, so daß die Mitarbeiter die Vorteile des zu erstellenden Softwaresystems erkennen, dürfte sich der Aufwand den Widerstand zu überwinden in Grenzen halten[43].

- Die **Unternehmenspolitik**, die hinter dem zu erstellenden Softwaresystem steht, ist die Erhöhung der Produktivität durch Optimierung des Arbeitsprozesses. Die Mitarbeiter und Angestellten sollen durch die zu erstellende Software in die Lage versetzt werden effektiver und effizienter unter Berücksichtigung ihrer Privatsphäre miteinander zu kooperieren.

[43] vgl. [GrPa95]

4.2 Der Informationsstandpunkt

Die Konzepte des Informationsstandpunktes bilden einen Rahmen für die Spezifikation der zu verarbeitenden Information im zu erstellenden Softwaresystem.
Die Konzepte bestehen aus Informationsobjekten, also Objekten, die Informationen, die das Softwaresystem betreffen, beinhalten, Beziehungen zwischen diesen Objekten und Verpflichtungen, bzw. Regeln, die das Informationsverhalten der Objekte beschreiben.

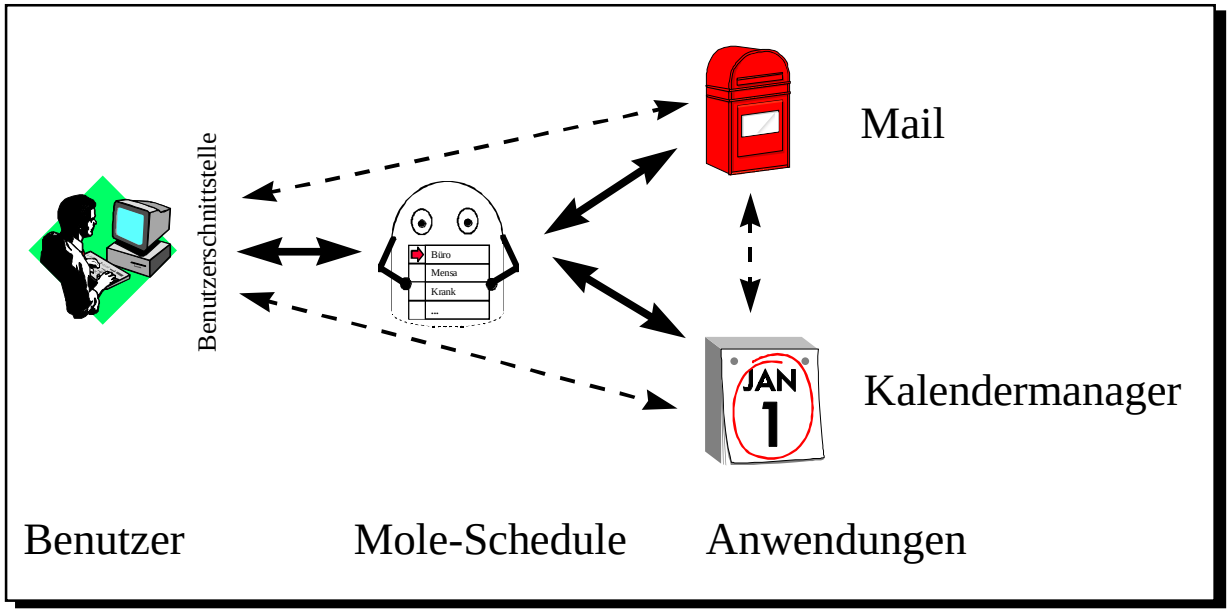

Abb.6: Beziehungen zwischen Informationsobjekten

Im wesentlichen existieren in dem hier zu erstellenden Softwaresystem vier Informationsobjekte:

1. Die **Benutzerschnittstelle** je Benutzer, bzw. je Anwender des Softwaresystems, also für jeden Mitarbeiter der Abteilung IPVR der Fakultät Informatik der Universität Stuttgart, für die Ein- und Ausgabe der benutzerspezifischen Daten, wie zum Beispiel der momentane Aufenthaltsort.

2. Das **Kalendermanagementwerkzeug** CM für die Verwaltung und Koordination der eingetragenen Termine oder Aufgaben.

3. Das **Mailtool** für die Verwaltung der elektronischen Post.

4. Die zu erstellende Software, namens **Mole-Office**, die die benutzerspezifischen Informationen verwaltet.

Beziehungen zwischen den Informationsobjekten

Der Anwender hat zum einen wie bisher die Möglichkeit, das Kalendermanagementwerkzeug CM und das Mailtool direkt über sein X-Windowssystem zu aktivieren und dort entsprechend Termine einzutragen und elektronische Post zu verschicken, zu lesen oder zu löschen.
Da aber ein integrierender Ansatz in dieser Arbeit verfolgt wird, d.h. am Ende der Entwicklung des Softwaresystems eine Anwendung entstehen soll, die alle verfügbaren Kommunikations-, bzw. Kooperationswerkzeuge enthält kann der Benutzer CM und das Mailtool auch indirekt über die **Benutzerschnittstelle** in Mole-Office aufrufen.
Hierbei kann der Benutzer dem Mailtool über die Benutzerschnittstelle die verschiedenen Adressaten der elektronischen Post angeben. Der Benutzer erhält außerdem vom Mole-Office-System direkt über die Benutzeroberfläche Informationen über andere Benutzer: inwieweit und wann sie sich an ihren Arbeitsplätzen oder in anderen Räumen oder sonstigen Örtlichkeiten aufhalten. Desweiteren wird angezeigt, ob andere entfernte Benutzer eine Terminanfrage per elektronischer Post an den lokalen Benutzer gestellt haben. Es können außerdem diese von Mole-Office automatisch erzeugten elektronischen Postnachrichten gelöscht, bzw. gespeichert werden.

Mole-Office speichert die benutzerspezifischen statischen und dynamischen Informationen. Es stellt aufgrund von Benutzereingaben und der Informationen aus dem Kalendermanagementwerkzeug elektronische Türzettel zur Verfügung und filtert die elektronische Post auf der Suche nach Nachrichten aus dem Mole-Office-System, damit der Benutzer darauf reagieren kann. Mole-Office kann automatisch elektronische Post generieren, falls der angefragte Benutzer im Augenblick nicht zu erreichen oder nicht unterbrechbar ist[44].

Das **Kalendermanagementwerkzeug CM** stellt die Termininformationen, die der Benutzer unter Angabe von Vertraulichkeitsstufen eingegeben hat, sowohl dem Benutzer, als auch dem Mole-Office-System zur Verfügung. CM kann außerdem automatisch elektronische Post mit einer Termininformation generieren, die dann an spezifizierte Anwender gesendet werden kann.

Das **Mailtool** verwaltet für den Benutzer und für Mole-Office die elektronische Post. Es kann die Termininformationen, die von einem anderen Benutzer mittels seinem CM erzeugt wurden, per „Drag-and-Drop“-Verfahren in den elektronischen Kalender eintragen.

[44] Anm: Dies entspricht einer Anrufbeantworterfunktionalität.

4.3 Der Berechnungsstandpunkt

Die Informationsobjekte des zu erstellenden Softwaresystems interagieren untereinander, indem sie Nachrichten über Schnittstellen senden oder empfangen. Der Schwerpunkt beim Berechnungsstandpunkt liegt also auf der Definition der Schnittstellen des zu entwerfenden Softwaresystems. Die Funktionalität der Informationsobjekte, die bereits im Informationsstandpunkt zur Veranschaulichung angegeben wurden, wird hier um der Vollständigkeit genüge zu tun, noch einmal angegeben.

4.3.1 Die Aspekte der Benutzerschnittstelle

- Die anderen Benutzer sollen als graphisches Symbol oder als Bild auf der graphischen Benutzeroberfläche dargestellt werden. Der Status des Benutzers soll auf einen Blick erkennbar sein. Desweiteren sollen Terminanfragen anderer Benutzer am Bildschirm durch ein blinkendes Symbol dargestellt werden.

- Durch Klicken mit der Maus auf das Symbol eines anderen Benutzers soll eine Anfrage über den genauen Status des Benutzers in der Vergangenheit, der Gegenwart oder in der Zukunft ermöglicht werden. Das Ergebnis dieser Anfrage enthält unter Wahrung von Vertraulichkeitsstufen zumindest den Namen des angefragten Benutzers und wenn es die Vertraulichkeitsstufe erlaubt, auch die Raumnummer, die Telefonnummer, den augenblicklichen Status und, wenn gewünscht, den Status zu einer benutzerdefinierten Zeit, z.B. in der Zukunft am Tag X um 11:15 Uhr.

- Der Benutzer soll seinen Aufenthaltsort und seinen augenblicklichen Grad der Unterbrechbarkeit, sowie seinen Informationsbedarf über eine direkte Eingabe mit der Maus seinen Wünschen entsprechend verändern können, oder dies durch einen Eintrag in seinem Kalender automatisch auslösen.

- Der Benutzer soll das Setup[45], das ihm die Möglichkeit bietet sein System individuell zu konfigurieren, direkt über die Benutzerschnittstelle aufrufen können.

- Die Aufrufe des Kalender- und Terminabsprachewerkzeug Calendar Manager, des elektronische Post verarbeitenden Mailtools und der Hilfefunktion sollen direkt über die graphische Benutzeroberfläche möglich sein.

- Beim Aufruf des Mailtools über Mole-Office soll der Benutzer andere Mitarbeiter über die Benutzerschnittstelle per Mausklick spezifizieren können.

[45] siehe Kapitel 5.3 Der Verwaltungsbereich

- Elektronische Post anderer Mitarbeiter in Bezug auf das Mole-Office-System soll am Bildschirm angezeigt, gelöscht oder gespeichert werden können.

4.3.2 Die Aspekte der Schnittstelle zum Mailtool

- Die zu erstellende Software Mole-Office soll in der Lage sein, elektronische Post darauf hin zu überprüfen, ob sich Post von einem anderen Mole-Office-Benutzer mit Termininformationen in dem Posteingangsordner des Benutzers befindet. Das Zeitintervall zur Überprüfung der Post soll vom Benutzer festgelegt werden können.

- Der Anwender soll in der Lage sein, automatisch erzeugte Post von einem anderen Mole-Office-Anwender von Mole-Office aus, nachdem er die Nachricht gelesen hat, zu löschen und zu speichern.

- Falls ein Anwender nicht erreichbar ist, soll Mole-Office automatisch, aber nach Bestätigung des Benutzers, elektronische Post erzeugen können, damit der gerufene über die Anfrage informiert wird.

- Damit die Kompatibilität zu zukünftigen Anwendungen gewährleistet ist, sollte die Schnittstelle zum Mailtool kompatibel zum Internet Mail Access Protocol (IMAP) und zum Simple Mail Transfer Protocol (SMTP) sein.

4.3.3 Die Aspekte der Schnittstelle zum Kalendermanagementwerkzeug CM

- Der Benutzer soll CM direkt von Mole-Office aus aufrufen können.

- Mole-Office soll alle Termininformationen unter Wahrung von Vertraulichkeitsstufen aus dem CM abfragen können. Diese müssen allerdings explizit vom Anwender angegeben, bzw. eingetragen werden.

- Damit die Kompatibilität zu zukünftigen Anwendungen gewährleistet ist, sollte die Schnittstelle zum CM kompatibel zum Internet Calendar Access Protocol (ICAP) sein.

4.3.4 Die Aspekte der Funktionalität von Mole-Office

- Der Benutzer soll beim ersten Anmelden an das System eine Art von Setup[46] durchlaufen, indem er seine benutzerspezifischen Angaben macht.

- Diese werden beim wiederholten Starten oder Anmelden an das System nicht wieder abgefragt, da sie gespeichert werden.

- Die benutzerspezifischen Angaben innerhalb des Setups, die beim ersten Anmelden gemacht wurden, können mittels des von der Benutzerschnittstelle aufrufbaren Setups jederzeit verändert, bzw. an die momentane Situation angepaßt werden.

- Der Status des Benutzers, also Ort, Zeit, Kooperationsbereitschaft und Interessenskontext soll in Abhängigkeit von Benutzereingaben über die Benutzeroberfläche und von Eingaben im elektronischen Kalender vom zu programmierenden Softwaresystem verwaltet werden.

- Wenn ein Benutzer nicht erreichbar ist, sollen andere Benutzer die Möglichkeit haben, über das Mole-Office automatisch eine Mail, mit einem vordefiniertem Text, generieren und an den entsprechenden Benutzer senden zu lassen, damit dieser über die Anfrage informiert wird. Der Benutzer sollte hierbei die Möglichkeit haben das Betreff-Feld dieser automatisch erzeugten Mail seinen Vorstellungen entsprechend auszufüllen.

- Ein Benutzer soll beim Aufrufen des zu programmierenden Softwaresystems Mole-Office automatisch über evtl. vorliegende elektronische Post, bzw. Anfragen von anderen Benutzern und ihren Benutzeragenten informiert werden.

- Er soll außerdem die Wahl haben die Anfragen von anderen Benutzern, seine Anwesenheit betreffend, automatisch nach dem Durchlesen aus seinem Mailordner entfernen bzw. speichern zu lassen.

[46] siehe Kapitel 5.3 Der Verwaltungsbereich

4.4 Der Engineeringstandpunkt

Im Folgenden wird versucht die Infrastuktur des zu erstellenden Softwaresystems, d.h. die Mechanismen, Askpekte und Funktionen, die für die Ausführung und Interaktion zwischen den verschiedenen Systemen benötigt werden, näher zu definieren.

- Eine mittlere Portabilität der Softwarekomponenten wird verlangt, d.h. die zu erstellende Software muß sowohl unter Solaris Version 2.4, als auch unter Windows NT Version 3.51 bzw. Windows 95 funktionieren.

- Die zu erstellende Software muß die Richtlinien zur Programmierung von Software vom 21.03.1994 der Abteilung IPVR einhalten. Dadurch wird ein von der Abteilung IPVR gewünschter Grad an Lesbarkeit und Dokumentierung des Codes, sowie ein gewünschter Grad an Einfachheit der Wartung erreicht.

- Der Benutzer des Systems soll die Möglichkeit haben mobile Agenten zur Überwachung von Aktivitäten eines anderen Benutzers und für die Informationsgewinnung einsetzen zu können.

- Das zu entwickelnde System soll zwischen unterschiedlichen Wahrscheinlichkeiten bei der Festlegung von Terminen, wie z.B. „findet garantiert statt“ oder „findet möglicherweise statt“, unterscheiden können. Diese Wahrscheinlichkeiten müssen vom Anwender in das Kalendermanagementwerkzeug CM explizit eingetragen werden.

- Die Änderung des Status eines Benutzers soll sofort oder nach benutzerdefinierbaren Intervallen angezeigt, bzw. abgefragt werden.

- Der Benutzer soll die Möglichkeit haben sich einen eigenen Aufenthaltsort zusätzlich zu den im System vorhandenen Standardaufenthaltsorten zu definieren. Die Standardaufenthaltsorte ergeben sich aus den Türschildern und Örtlichkeiten der Abteilung IPVR der Fakultät Informatik.

- Das System soll Anfragen anderer Benutzeragenten registrieren und, wenn gewünscht, am Bildschirm anzeigen. Gegebenenfalls soll es diese unter Wahrung von zusätzlich zu dem im CM definierten Vertraulichkeitsstufen beantworten. Diese Vertraulichkeitsinformationen müssen explizit vom Anwender im Kalendermanagementwerkzeug CM zu den Termininformationen eingetragen werden.

- Das System, das permanent läuft, kann sich, wenn der Benutzer es im Setup spezifiziert, die durchschnittlichen Arbeitsanfangs-, Arbeitsend-, und Mittagszeiten merken, bzw. errechnen. Mit diesen Daten kann das System dann wahrscheinliche Gründe für die Abwesenheit des Benutzers angeben, auch wenn dieser keine Eintragungen in das Kalendermanagementwerkzeug vorgenommen hat.[47]

- Das System soll eine Hilfefunktion beinhalten, mit der sich die Benutzer jederzeit eine Hilfe über die verfügbaren Funktionen geben lassen können.

- Die Informationen, die ein Benutzer über sich und seinen Zustand in das System eingegeben hat sollen persistent sein, d.h. sie sollen auch einen Ausfall des Arbeitsplatzrechners überstehen.

- Die Verwaltung des Status eines Benutzers soll auch von fremden Arbeitsplatzrechnern aus möglich sein. Dazu soll ein Paßwort zur Authentifizierung für jeden Benutzer eingerichtet werden.

4.5 Der Technologiestandpunkt

Im folgenden wird der Aspekt der Programmierumgebung und der zu verwendenden Komponenten untersucht:

- Die Programmierumgebung bilden SUN SPARCstations 10 mit dem Betriebssystem Solaris Version 2.4 und dem Windowsmanager Solaris Open Windows 3.4 von der Firma SUN Microsystems.

- Die Programmiersprache der zu erstellenden Softwarekomponenten ist Java Version 1.0.2, eine von der Firma SUN Microsystems entwickelte, objektorientierte Programmiersprache, die aufgrund ihrer Architektur eine plattformunabhängige Programmierung erlaubt.

- Bei dem zu verwendenden Agentensystem handelt es sich um Mole in der Version Alpha 1.0. Dies ist ein Agentensystem, das an der Universität Stuttgart entwickelt wurde und das mobile Agenten unterstützt.

- Als kooperationsunterstützende Softwarekomponenten sind Calendar Manager Version 3.3, ein von der Firma SUN Microsystems in das Betriebssystem Solaris Version 2.x

[47] Die Vorhersagegenauigkeit des Systems steigt proportional mit seiner Benutzung.

integriertes Kalender- und Terminabsprachewerkzeug, und Mailtool Version 3.3, ein ebenfalls von der Firma SUN Microsystems entwickeltes und in das Betriebssystem integriertes Werkzeug um elektronische Post zu verarbeiten, zu verwenden.

- Der Netscape Navigator Version 3.0b5a von der Firma Netscape Communications Corporation stellt den graphischen Rahmen dar, der für das Frontend zum Benutzer verwendet werden soll.

Nachdem die zu erstellende Software anhand der verschiedenen Untersuchungsstandpunkte, wie der Unternehmens-, der Informations-, der Berechnungs-, der Engineering- und der Technologiestandpunkt, von allen Seiten aus untersucht und spezifiziert wurde, folgt nun der Entwurf des Softwaresystems mit Hilfe des USCM-Modells.

5. Entwurf des Softwaresystems Mole-Office

Für den Entwurf des Softwaresystems Mole-Office wird das USCM-Modell [ChaMon95] verwendet. Das USCM-Modell (Universal Service Component Model) stellt ein Klassifikationsschema für den Entwurf eines Softwaresystems, das auch als ein Modell für die Interaktion der einzelnen Komponenten des Softwaresystems gesehen werden kann, zur Verfügung.
Es kann auch als eine Art Checkliste begriffen werden, die die Vollständigkeit des Entwurfs für das zu erstellende Softwaresystem garantiert. Bei der Anwendung des USCM-Modells wird das zu erstellende Softwaresystem in verschiedene, voneinander getrennte, wohlgeformte und definierte Komponenten, bzw. Funktionen zerlegt, die die spätere Wiederverwendung der Software erleichtern sollen. Im wesentlichen werden alle Dienste[48] des zu erstellenden Softwaresystems in eine Kernfunktion (Systemkern) und in eine die Kernfunktion umgebende Zugriffsschicht unterteilt. Die Zugriffsschicht gliedert sich im weiteren in drei Bereiche: den Gebrauchsbereich, den Verwaltungsbereich und den substantiellen Bereich.

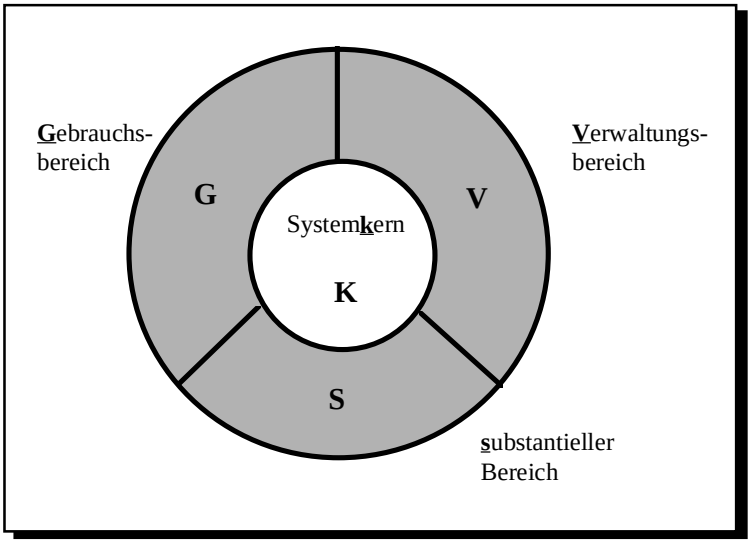

Abb.7: Die primäre Struktur des USCM-Modells

Die vier grundlegenden Bereiche werden wie folgt definiert:

Im **substantiellen Bereich** werden gewöhnlich die Benutzerschnittstellen und die zugrundeliegenden Konzepte der Kommunikation mit dem Benutzer definiert. Die Benutzer erhalten, wie die Entitäten im Gebrauchsbereich, eine Schnittstelle zum Systemkern. Durch diese Schnittstelle wird das Verhalten der Benutzer aus der Sicht des Systemkerns eingegrenzt und normalisiert.

[48] Bem.: Der Terminus „Dienst“ wird hier im Sinne der durch das zu erstellende Softwaresystem repräsentierten Funktion verwendet.

Im **Gebrauchsbereich** werden die Schnittstellen für die externen Komponenten oder Dienste, die den Systemkern direkt beeinflussen, angegeben. Die Entitäten in der externen Umgebung des zu erstellenden Softwaresystems, hier CM, Mailtool und Mole, erhalten somit einerseits eine (Server-) Schnittstelle zum Systemkern. Andererseits erhalten die Komponenten des Systemkerns eine normalisierte und definierte Sicht auf die Komponenten der externen Umgebung.

Die Aspekte des **Verwaltungsbereichs** stellen die logischen Datenstrukturen für die Initialisierung, die Konfiguration und andere Verwaltungsfunktionen, die gebraucht werden um das zu erstellende Softwaresystem einzurichten, bzw. zu konfigurieren, dar. Es wird eine Art von Verwaltungsschnittstelle definiert, die sowohl die Sicht von außen auf den Systemkern, als auch die Sicht vom Systemkern auf die externe Umgebung normalisiert.

Die Funktionen im **Systemkern** bilden die Logik für die Zusammenarbeit der anderen Bereiche, die in der Zugriffsschicht liegen. Sie sollten unabhängig von der externen Umgebung die Kernfunktionalität des zu erstellenden Softwaresystems zur Verfügung stellen.

Im folgenden werden die einzelnen Komponenten anhand des USCM-Modells genauer untersucht.

5.1 Der substantielle Bereich

Im substantiellen Bereich werden die zugrundeliegenden Konzepte der Kommunikation anhand des Status des Benutzers in Mole-Office erläutert.

Unter dem **Status eines Benutzers** werden

- die objektive Kooperationsbereitschaft, die von den im Kalendermanagementwerkzeug eingetragenen Terminen und Aufgaben, sowie von dem augenblicklichem Aufenthaltsort des Benutzers bestimmt wird,
- die subjektive Kooperationsbereitschaft, die der Benutzer über die Benutzeroberfläche direkt angeben kann, und
- der Interessenskontext des jeweiligen Benutzers zu einer gewissen Zeit verstanden.

Bevor die obigen Begriffe näher erklärt werden, müssen zunächst einige Grundlagen erläutert werden:

Die **Priorität einer Aktivität** wird zusätzlich zu den normalen Termininformationen in den CM für jeden Termin eingetragen. Die Priorität ist ein subjektiv wahrgenommener Einflußfaktor auf die Kooperationsbereitschaft. Es werden zwei mögliche Ausprägungen vorgeschlagen:

Normale, bzw. keine Priorität einer Aktivität:
Hierbei wird keine zusätzliche Termininformation in den CM eingetragen.

Priorität, bzw. hohe Priorität einer Aktivität:
Das Symbol für eine Aktivität mit hoher Priorität sind zwei Ausrufezeichen („!“). Diese werden zusätzlich in die Terminbeschreibung im CM eingetragen

Die **Standardaufenthaltsorte** in Mole-Office sind stark abteilungsabhängig. Bei den Standardaufenthaltsorten der Mitarbeiter der Abteilung IPVR der Fakultät Informatik der Universität Stuttgart handelt es sich um:[49]

- das eigene Büro,
- die anderen Büros der Mitarbeiter der Abteilung,
- die Abteilungen von anderen Mitarbeitern der Fakultät Informatik,
- die Besprechungsräume,
- die Diplomandenarbeitsräume (IPVR-, AS-Pool)
- die Hörsäle und Seminarräume ,
- die Bibliothek,
- die verschiedenen Kopiergeräte,
- die Mensa, bzw. „beim Essen“,
- die Kommunikationsecke,
- zu Hause,

[49] vgl. Türschilder in der Abteilung IPVR der Fakultät Informatik an der Universität Stuttgart

- im Urlaub,
- „im Bett“, bzw. „krank“,

- externes Treffen, bzw. Dienstreise

- und benutzerdefinierte Aufenthaltsorte.

Zu jedem Ort werden die entsprechende Raumgröße, die Telefonnummer und die vorhandenen speziellen Ausrüstungsgegenstände, wie z.B. ein Overheadprojektor, eine Tafel oder Sonstiges angeben. Die Standardaufenthaltsorte der Abteilung sind fest in das Softwaresystem integriert. Erweiterungen um benutzerdefinierte Aufenthaltsorte werden von den Systemagenten verwaltet.
Zusätzlich ist zu jedem Standardaufenthaltsort eine „**Standardkooperationsbereitschaft**“ angegeben, die in die Berechnung des Status des Benutzers einfließt. Hierbei wird davon ausgegangen, das der Mitarbeiter nur in seinem eigenen Büro die höchste Kooperationsbereitschaft in bezug auf andere Mitarbeiter hat. Befindet sich der Mitarbeiter an einem anderen Aufenthaltsort, so ist seine Kooperationsbereitschaft von den anderen Mitarbeitern aus gesehen prinzipiell niedriger als in seinem eigenem Büro. Der Benutzer kann diese Standardkooperationsbereitschaft über das dem Softwaresystem integrierte Setup entsprechend seine Wünschen und Vorstellungen anpassen.

Def.: **Kooperationsbereitschaft**:

Unter Kooperationsbereitschaft wird im folgenden der Wille eines Mitarbeiters zur Kooperation und sein daraus resultierendes Verhalten in Bezug auf Störungen zu einem bestimmten Zeitpunkt verstanden.

Es gibt zwei grundlegende Einflußfaktoren auf die Kooperationsbereitschaft der Benutzer: objektive und subjektive.

Die **objektiven** Einflußfaktoren auf die Kooperationsbereitschaft sind die Termine und Aufgaben, die der Mitarbeiter abzuarbeiten hat, also die Termininformationen aus dem Kalendermanagementwerkzeug CM des Benutzers, wobei hier unterschieden wird in:
- kein(e) Termin/Aufgabe in den Kalender eingetragen,
- ein(e) Termin/Aufgabe ohne Priorität in den Kalender eingetragen,
- ein(e) Termin/Aufgabe mit Priorität in den Kalender eingetragen, und
- keine Information aus dem Kalender, da der Termin, bzw. die Aufgabe einer Vertraulichkeitsstufe zugeordnet ist, die höher als die eigene Stufe in Bezug auf den Benutzer ist[50].

[50] Siehe Kapitel 5.3 Der Verwaltungsbereich

Der zweite objektive Einflußfaktor auf die Kooperationsbereitschaft ist der Standardaufenthaltsort des Benutzers. Der Benutzer kann seinen Aufenthaltsort entweder über die Benutzeroberfläche, oder über einen zusätzlichen Eintrag in seinen Kalender angeben. Hierbei wird unterschieden in:

- der Standardaufenthaltsort und die entsprechende Kooperationsbereitschaft, und
- andere Aufenthaltsorte und die entsprechenden Kooperationsbereitschaft.

Die beiden objektiven Einflußfaktoren werden in Kombination miteinander verwendet, d.h. z.B. ein Benutzer hat keinen Eintrag in seinem Kalender und befindet sich in seinem Büro, oder er hat einen Termin mit Priorität und befindet sich im Büro seines Vorgesetzten.

Die **subjektiven** Einflußfaktoren auf die Kooperationsbereitschaft eines Mitarbeiters sind:

- Die persönliche Einschätzung der jeweiligen Arbeitssituation, die der Mitarbeiter über die Benutzeroberfläche direkt angeben kann.
- Die in den Terminkalender eingetragenen Prioritäten und die Kooperationsbereitschaft des Mitarbeiters in Bezug auf die einzelnen Aufgaben, und
- die im Setup evtl. geänderte Kooperationsbereitschaft der Standardaufenthaltsräume.

Die aktuelle Kooperationsbereitschaft des Anwenders wird prinzipiell aus den objektiven und subjektiven Einflußfaktoren berechnet, wobei die subjektiven Faktoren gegenüber den objektiven Faktoren bevorzugt werden, da der Anwender explizite Angaben über seine Kooperationsbereitschaft gemacht hat.
Hat der Anwender z.B. einen Termin mit Priorität in seinen CM eingetragen, so ist seine Kooperationsbereitschaft laut Softwaresystem niedrig. Will der Anwender aber eine hohe Kooperationsbereitschaft während des Termin, so kann er entweder den entsprechenden Knopf auf der Benutzeroberfläche drücken oder zusätzliche Kooperationsinformationen in seinen Kalender eintragen. Definiert er seine momentane Kooperationsbereitschaft mittels den Knöpfen auf der Benutzeroberfläche um, so ist diese Änderung nur für die Dauer der augenblicklichen Aktivität. Ist diese beendet, so berechnet das Softwaresystem die momentane Kooperationsbereitschaft wieder aus den objektiven Einflußfaktoren.

Die Berechnung der aktuellen Kooperationsbereitschaft vom Benutzer durch das Softwaresystem aufgrund der objektiven Einflüsse wird durch folgende Abbildung veranschaulicht, in der die Kooperationsbereitschaft in drei Stufen mit unterschiedlicher Unterbrechbarkeit[51] unterteilt wird:

Objektiver Einfluß / Kooperations-	anwesend am Arbeitsplatz				nicht anwesend am Arbeitsplatz			
	kein Termin	Termin ohne Priorität	Termin mit Priorität	keine Information aus Kalender	kein Termin	Termin ohne Priorität	Termin mit Priorität	keine Information aus Kalender
1. Stufe: kontaktfreudig	✓							
2. Stufe: „Bitte nicht stören“		✓			✓	✓		
3. Stufe: „Keine Störung“			✓	✓			✓	✓

Abb.8: Die objektiven Einflußfaktoren auf die Kooperationsbereitschaft der Benutzer

1. Stufe: Kontaktfreudig, bzw. hohe Kooperationsbereitschaft

Der Benutzer ist zur Kooperation bereit, d.h.

⇒ er befindet sich im Augenblick an seinem Arbeitsplatz und er hat keinen Termin und keine Aufgabe im CM eingetragen, oder

⇒ er hat die subjektive Einflußfaktoren entsprechend geändert.

⇒ Die Unterbrechbarkeit des Benutzers ist hoch.

⇒ Will der Benutzer einen Termin mit hoher Kooperationsbereitschaft in seinem Kalender angeben, so muß er das Sonderzeichen „@“ in die Terminbeschreibung eintragen.

[51] Anmerkung: Da die Definition von Unterbrechbarkeit von Individuum zu Individuum verschieden ist, wird hier Unterbrechbarkeit, als die Größe der Störung des Arbeitsablaufes eines Mitarbeiters gesehen.

2. Stufe: „Bitte nicht stören...“, bzw. mittlere Kooperationsbereitschaft

Der Benutzer ist eingeschränkt zur Kooperation bereit, d.h.

⇒ er befindet sich an seinem Arbeitsplatz und hat einen Termin ohne Priorität in seinen Kalender eingetragen, oder

⇒ er befindet sich nicht an seinem Arbeitsplatz und hat keinen Termin in seinen Kalender eingetragen, oder

⇒ er befindet sich nicht an seinem Arbeitsplatz und hat einen Termin ohne Priorität in seinen Kalender eingetragen, oder

⇒ er hat die subjektive Einflußfaktoren entsprechend geändert.

⇒ Die Unterbrechbarkeit des Benutzers ist "mittel".

⇒ Will der Benutzer einen Termin mit mittlerer Kooperationsbereitschaft in seinen Kalender angeben, so muß er die Sonderzeichen „@@“ in die Terminbeschreibung eintragen.

3. Stufe: „Keine Störung!“, „Status unbekannt!“, bzw. niedrige Kooperationsbereitschaft

Der Benutzer ist nicht zur Kooperation bereit, d.h.

⇒ er hat einen Termin mit Priorität in seinen Kalender eingetragen, oder

⇒ er hat einen Termin mit zusätzlicher Vertraulichkeitsinformation in seinen Kalender eintragen, und der Mitarbeiter ist einer niedrigeren Vertraulichkeitsstufe zugeordnet.

⇒ er hat die subjektive Einflußfaktoren entsprechend geändert.

⇒ Der Benutzer ist, seines Ermessens nach bis auf Notfälle, nicht unterbrechbar.

⇒ Will der Benutzer einen Termin mit niedriger Kooperationsbereitschaft in seinen Kalender angeben, so muß er die Sonderzeichen „@@@“ in die Terminbeschreibung eintragen.

Wie die Kooperationsbereitschaft läßt sich auch der Interessenskontext eines Benutzers abgestuft definieren:

Der **Informationsbedarf** bzw. der **Interessenskontext** der Benutzer definiert eine Menge von anderen Benutzern, bzw. potentiellen Kooperationspartnern und entsprechenden Ereignistypen[52] die für den jeweiligen Benutzer von Interesse sind. Der Informationsbedarf in Bezug auf einen anderen Benutzer im Mole-Office-System läßt sich in zwei unterschiedliche Stufen unterteilen:

1. Stufe: „hoher Informationsbedarf“

Der Anwender will über **alle** Ereignisse in Bezug auf einen Benutzer des Systems über die Benutzeroberfläche, oder per elektronischer Post informiert werden.
Das heißt im Interessenskontext des lokalen Anwenders sind alle diejenigen Benutzer eingetragen, an deren Kooperationsbereitschaft der Anwender interessiert ist.
Befindet sich der lokale Anwender nicht an seinem Arbeitsplatz, werden alle Ereignisse zwischengespeichert und beim Anmelden des Anwenders entsprechend der Spezifikation des Anwenders an ihm übermittelt.
Zusätzlich erhält der Anwender die vom Mole-Office-System automatisch generierte elektronische Post anderer Benutzer.

=> Der Anwender hat an alle anderen Kollegen von Interesse einen Agenten zur Beobachtung (Watch-Agent) gesendet. Diese übermitteln, innerhalb der entsprechenden Vertraulichkeitsstufe, alle Ereignisse an den Systemagenten.

2. Stufe: „kein Informationsbedarf“

Der Anwender will nicht über die Ereignisse eines anderen Benutzers im Mole-Office-System informiert werden, d.h. die Kooperationsbereitschaft des entsprechenden Benutzers wird nicht an der Benutzeroberfläche dargestellt.
Alle Ereignisse von diesem Benutzers werden für den lokalen Anwender nicht zwischengespeichert.
Der lokale Anwedner erhält nur die vom Mole-Office-System automatisch generierte elektronische Post.

=> Der Anwender hat keinen Watch-Agenten an den versandt.

Der Interessenskontext eines Benutzers enthält also eine Menge von anderen Benutzern,

[52] siehe Kapitel 5.4 Der Systemkern

- von denen Ereignisse empfangen werden,
- deren Status innerhalb der Benutzeroberfläche dargestellt wird,

- und Vertraulichkeitsstufen[53] des jeweiligen Benutzers.

Nachdem die zugrundeliegenden Konzepte der Kommunikation erläutert wurden, wird nun die **Benutzerschnittstelle** des Mole-Office-Systems wie folgt gestaltet:

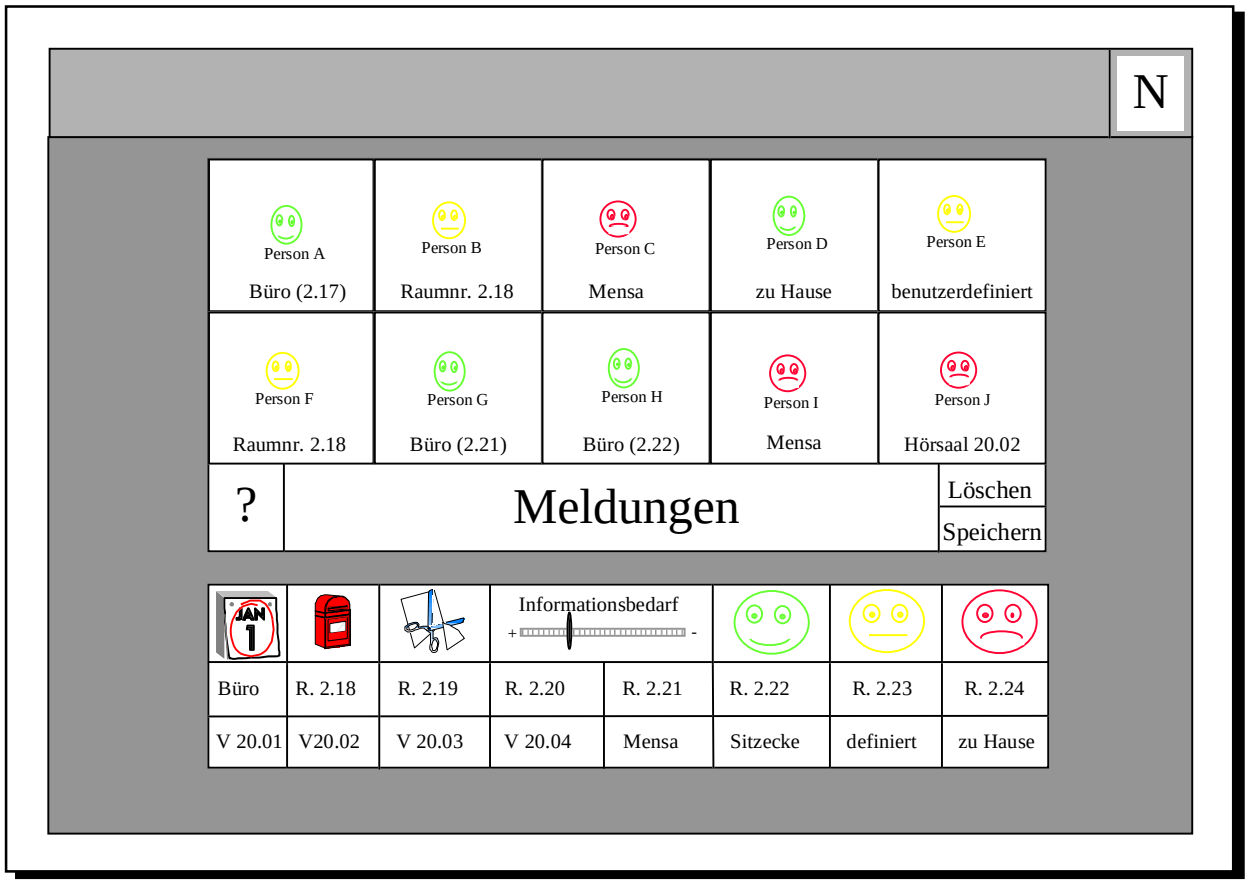

Abb.9: Der Entwurf der Benutzerschnittstelle

Die Benutzerschnittstelle gliedert sich in zwei Bereiche: den Informationsbereich und den Aktionsbereich.

Der **Informationsbereich**:

Im ersten, oberen Bereich, dem Informationsbereich, werden die im Setup von Mole-Office, bzw. die im Interessenskontext angegebenen Mitarbeiter der Arbeitsgruppe durch

[53] siehe Kapitel 5.3 Der Verwaltungsbereich

farbige Symbole eines Gesichtes, die die augenblickliche Kooperationsbereitschaft der jeweiligen Benutzer angeben, dargestellt.
Unter den Symbolen steht der jeweilige Namen des Mitarbeiters, und der Raum, bzw. der Ort an dem sich der Mitarbeiter im Augenblick befindet. Im Informationsbereich werden die Mitarbeiter-Ort-Beziehungen und die Kooperationsbereitschaft der ausgewählten Teamkollegen dargestellt.
Klickt ein Benutzer auf eines dieser Symbole kann der Benutzer genauere Informationen über den angewählten Benutzer vom Mole-Office-System erfragen. Dazu wird ein Eingabefeld geöffnet, in dem der Name des angewählten Benutzers, seine Telefonnummer, ein Datumsfeld mit dem aktuellen Datum, ein Zeitfeld mit der aktuellen Zeit angezeigt werden. Der Benutzer kann das Datum und die Zeitangabe nach seinen Wünschen verändern, um so z.B. den Status des anderen Benutzers in, oder vor zwei Tagen abzufragen und damit weitere Aktivitäten effizient planen, bzw. nachvollziehen zu können. Durch Drücken eines Bestätigungsknopfes wird die Anfrage gestartet, indem der Systemagent einen Ask-Agenten[54] erzeugt. Anschließend werden die Informationen über den Status des angewählten Benutzers in der Meldungsleiste dargestellt. Das kleine Eingabefeld verschwindet wieder. Drückt der Benutzer einen Abbruchknopf, so verschwindet das kleine Eingabefeld ebenfalls wieder.
Desweiteren hat ein Anwender die Möglichkeit über die Angabe eines benutzerspezifischen Paßwortes und Angabe eines Standardaufenthaltsortes den Aufenthaltsort eines bestimmten Benutzers zu verändern. Dadurch wird dem Anwender und den vom Anwender autorisieren Mitarbeitern ermöglicht ggf. auch von fremden Arbeitsplätzen aus den Standardaufenthaltsort des jeweiligen Anwenders zu verändern.

Startet ein anderer Benutzer eine Anfrage, oder hat er eine Mail vom Mole-Office-System erhalten, so blinkt das entsprechende Symbol des Anwenders. Klickt der Anwender auf dieses Symbol, so wird eine entsprechende Meldung, bzw. Mail in der Meldungszeile ausgegeben.

Unterhalb der Anzeige der Benutzer werden aktuelle Systemmeldungen, z.B. der Erhalt einer Mail von einem anderen Benutzer des Mole-Office-Systems, oder entsprechende abgefragte Benutzerinformationen einer Anfrage des Mole-Office-Systems angezeigt. Zusätzlich zu der Systemmeldungsleiste ist hier ein Hilfeknopf plaziert, mit dem sich der Benutzer die Hilfe zu den aktuell ausführbaren Funktionen anzeigen lassen kann und ein Löschen-, bzw. Speichern-Knopf, mit dem der Benutzer erhaltene Mails vom Mole-Office-System bei Bedarf löschen, bzw. speichern kann.

Die Benutzeroberfläche unterstützt sowohl die synchrone Kommunikation, durch die Anzeige des Status des Benutzers, also auch die asynchrone Kommunikation, z.B. durch das Anzeigen vom Erhalt einer elektronischen Nachricht mit Termininformationen.

[54] siehe Kapitel 5.2 Der Gebrauchsbereich

Die Anzeige ist nicht raumorientiert, d.h. im Informationsbereich werden die einzelnen Mitglieder der Arbeitsgruppe angezeigt, die man im Setup angegeben hat und nicht die Standardaufenthaltsorte der Arbeitsgruppe. Wäre die Benutzerschnittstelle raumorientiert, so wäre bei großen Versammlungen, wie z.B. bei der Gruppensitzung, keine Übersicht mehr vorhanden.

Der **Aktionsbereich**:

Den zweiten, unteren Bereich der Benutzeroberfläche stellt der Aktionsbereich des Benutzers dar. Hier kann der Benutzer das Kalendermanagementwerkzeug CM, das Mailtool und das Setup[55] aufrufen, indem er auf die entsprechenden Knöpfe drückt.

Drückt der Benutzer den Mail-Knopf, so öffnet sich ein Mail-Fenster mit einem Empfängerfeld, einem Subjectfeld, bzw. Betreff-Feld, einem Abbruch-Knopf, einem Sende-Knopf und einem to-Mail-Knopf über dem Aktionsbereich.
Der Benutzer kann die verschiedenen Empfänger, per Mausklick auf die entsprechenden Symbole, in das Empfängerfeld einfügen., Damit das Mole-Office-System entsprechende Mails erkennen kann, steht an erster Stelle des Betreff-Felds ein automatisch generierter Text. Durch Drücken eines Sende-Knopfes wird die Mail verschickt und das Mail-Fenster geschlossen. Wird ein Abbruch-Knopf gedrückt, wird das Mail-Fenster wieder geschlossen. Wird ein to-Mail-Knopf gedrückt, so wird das Mailtool aufgerufen und das Mail-Fenster ebenfalls wieder geschlossen.

Will der Benutzer die Angaben des Setups ändern, so muß er den Setup-Knopf drücken. Ein Setup-Fenster mit Eingabefeldern für jede benutzerspezifische Information wird am Bildschirm dargestellt. Im Setup-Fenster gibt es außerdem Knöpfe für den Interessenskontext eines Mitarbeiters, das Speichern der Änderungen mittels eines Speichern-Knopfes und das Verwerfen der Änderungen mittels eines Abbruch-Knopfes.

Der Benutzer kann desweiteren seinen Informationsbedarf, bzw. seinen Interessenskontext, durch Anklicken des Interessenskontext-Knopfes mit der Maus aktivieren. Dieser befindet sich sowohl im Setup-Fenster, als auch direkt im Aktionsbereich. Beim Anklicken öffnet sich ein Info-Fenster, in dem der Benutzer seinen Interessenskontext angeben kann:

- Die Mitarbeiter, die im Informationsbereich dargestellt werden.
- Die Vertraulichkeitsstufen für alle angewählten Mitarbeiter.

Hat der Anwender seine Änderungen im Interessenskontext abgeschlossen, so hat er die Möglichkeit seine Änderungen zu speichern oder zu verwerfen.

[55] siehe Kapitel 5.3 Der Verwaltungsbereich

Der Anwender kann seine subjektive Kooperationsbereitschaft mittels Anklicken eines der farbigen Symbole, bzw. Gesichter temporär verändern. Hierbei hat die so gewählte Kooperationsbereitschaft Priorität gegenüber der vom System berechneten. Die Änderungen bleiben nur für den augenblicklichen Termin, oder Ort bestehen. Wechselt der Anwender seinen Aufenthaltsort, bzw. endet dieser Termin, oder es beginnt ein neuer Termin, wird also ein Ereignis im Mole-Office-System ausgelöst, wird die Kooperationsbereitschaft wieder an vom Softwaresystem berechnete angepaßt:

- Das **grüne, lächelnde** Gesicht steht für eine hohe Kooperationsbereitschaft, also für die erste Stufe „Kontaktfreudig“ des zuvor erklärten Modells.
- Das **gelbe, neutrale** Gesicht steht für eine mittlere Kooperationsbereitschaft. Diese entspricht der zweiten Stufe „Bitte nicht stören...“.
- Das **rote, ärgerliche** Gesicht steht schließlich für eine niedrige Kooperationsbereitschaft und entspricht der dritten Stufe „Keine Störung“.

Der Anwender kann insbesondere seine augenblickliche Kooperationsbereitschaft direkt von der Benutzeroberfläche über das entsprechend ausgewählte Gesichtssymbol ablesen.

Indem der Benutzer einen der Raumknöpfe drückt, kann er sich von Standardaufenthaltsort zu Standardaufenthaltsort bewegen, um z.B. ein spontanes Treffen mit einem Mitarbeiter an einem entsprechenden Standardaufenthaltsort abzuhalten.

5.2 Der Gebrauchsbereich

In diesem Bereich werden die Schnittstellen zu den externen Komponenten wie Mailtool und Kalendermanagementwerkzeug CM definiert. Desweiteren wird hier die Einbindung von Mole-Office in das Agentensystem Mole erläutert.

Das zu erstellende Werkzeug Mole-Office besteht aus verschiedenen, individuell konfigurierbaren Objekten[56] für jeden Benutzer und der Kommunikation zwischen diesen Objekten.
Bei diesen Objekten, bzw. Komponenten handelt es sich um:[57]

- Die Benutzerschnittstelle, um für den Anwender interessante Beziehungen darzustellen, um dem Anwender die Möglichkeit zu geben Informationen über seinen aktuellen

[56] siehe Kapitel 4.2 Der Informationsstandpunkt
[57] siehe Abb. 7

Status[58] in das System einzutragen und um die Werkzeuge CM und Mailtool integrieren zu können.

- Eine Location innerhalb von Mole für jeden Standardaufenthaltsort innerhalb der Abteilung IPVR an der Fakultät Informatik der Universität Stuttgart.

- Ein Systemagent pro Benutzer innerhalb einer Location von Mole, der alle Informationen verwaltet.

- Mobile Watch-Agenten, die von den jeweiligen Anwendern über die Systemagenten ausgeschickt werden, um die Aktivitäten eines speziellen Benutzers im gleichen Team unter der Wahrung von Vertraulichkeitsstufen durchgehend zu beobachten und Veränderungen an den eigenen Benutzer zurück zu melden.

- Mobile Ask-Agenten, die von den jeweiligen Anwendern über die Systemagenten an einen bestimmten Benutzer ausgeschickt werden, um spezielle Informationen über diesen Benutzer zu erfragen.

Die folgende Grafik verdeutlicht diesen Zusammenhang:

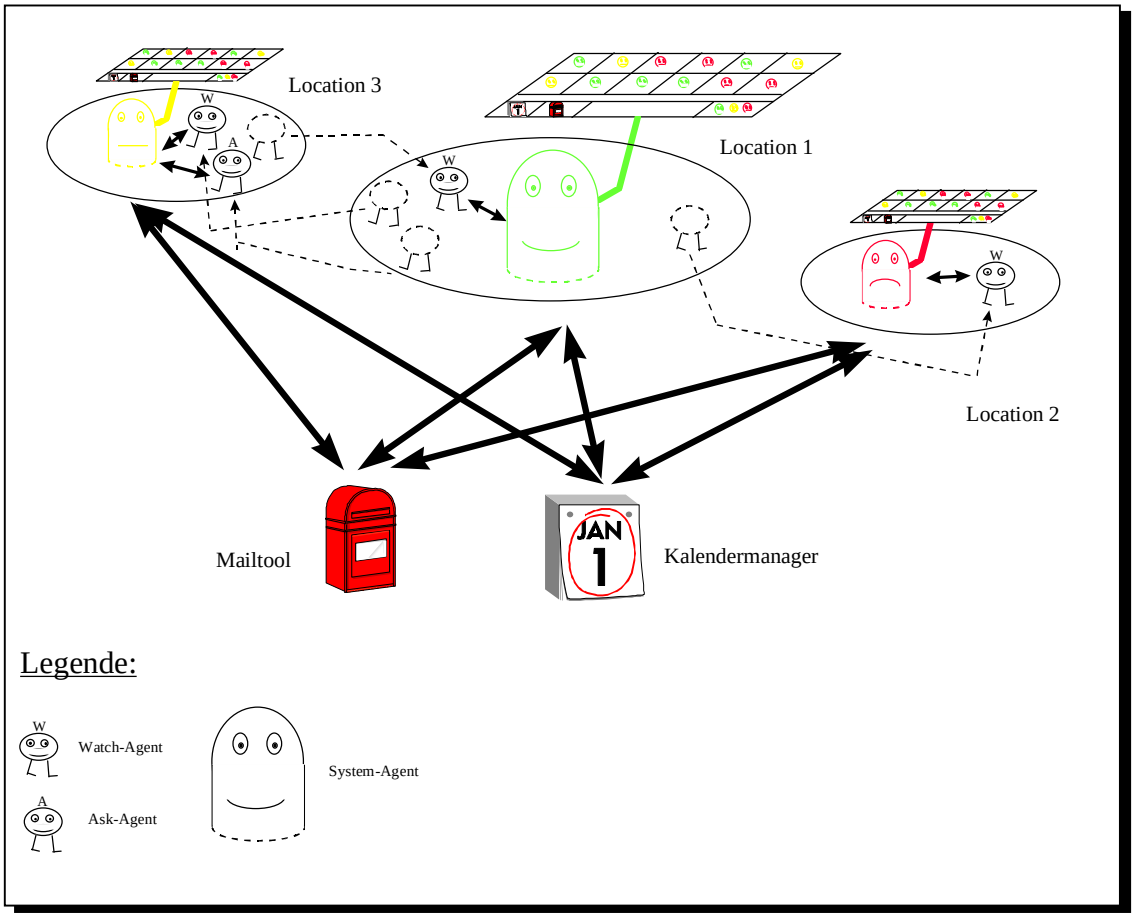

Abb. 10: Einbindung von Mole-Office in das Agentensystem Mole

[58] siehe Kapitel 5.1 Der substantielle Bereich

Einbindung von Mole-Office in Mole

Die Unterstützung für die Teamkoordination von Mole-Office für jeden Benutzer wird durch eine gewisse Anzahl von benutzerspezifischen Agenten an speziellen Orten, bzw. Locationen innerhalb von Mole realisiert. Jeder Standardaufenthaltsort in Mole-Office stellt eine **Location** innerhalb von Mole dar. Jeder Agent hat eine speziell ihm übertragene Aufgabe zu erfüllen:

Jeder Benutzer hat einen **ortsfesten Systemagenten** in seinem Standardbüro, der alle Informationen in Bezug auf den Benutzer verwaltet. Mittels des Systemagenten werden auch die anderen zu integrierenden Werkzeuge gestartet, bzw. auf Informationen durchsucht. Nur der Systemagent kann über die Benutzerschnittstelle mit dem Anwender kommunizieren und umgekehrt. Systemagenten werden über den Namensdienst im Agentensystem angesprochen.

Der Benutzer hat außerdem die Möglichkeit **mobile Agenten** einzusetzen, die Informationen zwischen den Systemagenten übertragen können. Die Anzahl von mobilen Agenten pro Benutzer ist nicht festgelegt.

Je nach spezifiziertem Interessenskontext sendet der lokale Systemagent einen mobilen **Watch-Agenten** zu dem entsprechenden Systemagenten des anderen Benutzern, um deren Aktivitäten dauerhaft, innerhalb der spezifizierten Vertraulichkeitsstufen, zu beobachten und Änderungen im Status des Benutzers, bzw. im Status des Systemagenten des Benutzers an den auftraggebenden Systemagenten zu melden.

Nachdem der Watch-Agent vom lokalen Systemagenten erzeugt wurde, migriert er zu den entsprechenden entfernten Systemagenten des zu beobachtenden Benutzers und versucht sich dort erstens zu authentifizieren, damit nur berechtigte Agenten des Mole-Office-Systems Informationen erhalten und zweitens sich in eine Informationsliste beim entfernten Systemagenten eintragen zu lassen. Der entfernte Systemagent entscheidet über Erfolg oder Nichterfolg dieses Eintrages anhand des Vertrauens, daß der anfragende Benutzer beim Benutzer des entfernten Systemagenten genießt. Ist der anfragende Watch-Agent vertrauenswürdig, so wird er in die Informationsliste eingetragen, ist er nicht vertrauenswürdig, so wird er abgewiesen und mit einer entsprechenden Meldung zurückgesendet. Wird der anfragende Watch-Agent in die Informationsliste aufgenommen, werden die auftretende Ereignisse an den anfragenden Systemagenten gesandt.

Die mögliche Verschwendung von Ressourcen des Computersystems, wenn man den schlechtesten Fall von nx(n-1) Watch-Agenten betrachtet, der dann eintritt, wenn in einem Team von n Mitarbeitern alle Mitarbeitern ihre (n-1) möglichen Kollegen in ihren Interessenskontext aufgenommen haben, kann damit ausgeglichen werden, daß die entsprechenden Informationen schon am Entstehungsort gefiltert werden und nur die minimale Information übertragen wird. Desweiteren wird durch diese Technik eine asynchrone und parallele Informationsverarbeitung ermöglicht.
Bei einzelnen Anfragen in Bezug auf Mitarbeiter außerhalb des Teams, des spezifizierten Interessenskontextes und wenn ein Benutzer den Update!-Knopf gedrückt hat, wird vom Systemagenten ein mobiler **Ask-Agent** erschaffen. Dieser muß im Gegensatz zu den Watch-Agenten nicht unbedingt zu dem entsprechenden Systemagenten migrieren um die gewünschten Informationen zu erhalten. Er entscheidet sich aufgrund eines Systemmechanismus[59] von Mole ob sich die Migration für ihn in diesem Fall lohnt, oder ob die Informationen über RPC´s günstiger zu besorgen sind. Liegen die Ergebnisse der Anfrage vor, werden sie an den auftraggebenden Systemagenten weitergeleitet. Ist die Aufgabe des Ask-Agenten erfüllt, so wird er zerstört.

Schnittstelle zum Kalendermanager

Wie bereits erwähnt kann das Kalendermanagementwerkzeug CM direkt über die Benutzerschnittstelle von Mole-Office aufgerufen werden. Es ist desweiteren möglich verschiedene Termininformationen aus einem angegebenen Kalender, unter Berücksichtigung von verschiedenen Vertraulichkeitsstufen, zu erhalten. Dazu stehen dem Systemagenten folgende standardisierte Befehle, die sich am Internet Access Protokoll[60] orientieren und die in CORBA IDL beschrieben werden, zur Verfügung:

```
interface CalenderManager{
        // Kalender zum Lesen öffnen
        void CALSELECT(
                in string name,       // Name des CM
                out integer result    // Ergebnis
        );




        // den Kalender wieder schließen
```

[59] vgl. [Voi96]
[60] vgl. [ICAP96]

```
        void CALCLOSE (
                in string name,       // Name des CM
                out integer result    // Ergebnis
        );

        // Zeitrahmen für eine Abfrage festlegen
        void CALRANGE (
                in string name,             // Name des CM
                in string start_date_time,  // Startzeit
                in string end_date_time,    // Endzeit
                out integer result          // Ergebnis
        );

        // dem Zeitrahmen entsprechende Aktivitäten auslesen
        void CALEXISTS (
                in string name,                     // Name des CM
                out string[] activity_description,  // Terminbeschreibung
                out integer result                  // Ergebnis
        );

        // Kalendermanager starten
        void CALSTART (
                in string name,       // Name des CM
                out integer result    // Ergebnis
        );
}
```

Wobei „name“ der Name des Terminkalenders ist. Die Zeitangaben werden im folgenden Format[61] an Mole-Office übergeben: „yyyymmddThhmmss“, mit:

yyyy => Jahreszahl (1900 - 9999)
mm => Monat (01-12)
dd => Tag (01-31)
T => eindeutiges Trennzeichen
hh => Stunde (01-24)
mm => Minute (01-60)
ss => Sekunde (01-60).

Die Ergebnisse, die in der Variablen „result“ zurückgegeben werden, sind Null, bzw. eins für „Operation erfolgreich“, bzw. „Operation nicht erfolgreich.
Die „CALSTART“-Funktion ist nicht innerhalb des ICAP-Protokoll definiert, sie wird aber hier für die Integration des Kalendermanagementwerkzeugs CM benötigt. Durch diese

Funktion erhalten die Anwender die Möglichkeit mit der gewohnten Benutzeroberfläche von CM zu arbeiten und dort ihre Termine und Aufgaben, entsprechend ihren bisherigen Gewohnheiten, einzutragen.

Schnittstelle zur Email

Um die asynchrone Kommunikation zwischen den Benutzern zu ermöglichen und um die Termine aus der elektronischen Post direkt in CM eintragen zu können, muß dem System(-Agent) die Möglichkeit gegeben werden auf ein Mailtool, das die elektronische Post für den Benutzer verwaltet, in irgendeiner Form zuzugreifen. Dazu werden folgende Funktionen[62], die wiederum im CORBA IDL beschrieben werden, zur Verfügung gestellt:

```
interface Mailtool {
        // Aufruf des Mailtools
        void MailStart (
                out integer result    // Ergebnis
        );

        // Mailbox auswählen und Check auf neue Mails
        void SELECT (
                in string Mailbox_name,    // Name der Mailbox
                out string Flags,          // Sind neue Nachrichten da?
                out integer result         // Ergebnis
        );
        // Nachrichten auslesen
        void FETCH (
                in string MailBox_name,    // Name der Mailbox
                out string subject,        // Subjektzeile einer Nachricht
                out string mail_header,    // Kopfzeile einer Nachricht
                out integer mail_number    // Nummer einer Nachricht
                out integer result         // Ergebnis
        // Erzeugen von Mails
        void NewMail (
                in string to_users,        // Liste der Empfänger
                in string subject,         // Subjektzeile der Nachricht
                in string mail_header,             // Kopfzeile der Nachricht
                in string message,         // Inhalt der Nachricht
```

[61] Anmerkung: Dieses Format entspricht dem ISO8601 Format.
[62] vgl. [Cri94]

```
            out integer result          // Ergebnis
        );
        // Löschen von Mails
        void DelMail (
            in string MailBox_name,     // Name der Mailbox
            in integer mail_number,     // Nummer der Mail
            out integer result          // Ergebnis
        );
}
```

Wobei der Parameter „result“ analog zum CalenderManager-Interface verwendet wird. Um eine Kompatibilität zu bestehenden Mailsystemen herzustellen wird zum Versenden der einzelnen Mails das Simple Mail Transfer Protocol (SMTP[63]) verwendet.

Der Anwender kann selbstverständlich auch andere Mailprogramme mit der entsprechenden Syntax der Befehle verwenden. Das einzige Problem hierbei ist, daß die „Drag-and-Drop“-Funktionalität in Bezug auf die elektronische Post, die vom Kalendermanagementwerkzeug erzeugt wird, verloren geht.

5.3 Der Verwaltungsbereich

Im Verwaltungsbereich wird auf die logischen Datenstrukturen für die Initialisierung, die Konfiguration und andere Verwaltungsfunktionen, die gebraucht werden um das zu erstellende Softwaresystem zu konfigurieren, näher eingegangen.

Der Verwaltungsbereich entspricht in weiten Teilen dem zu integrierendem **Setup** von Mole-Office, da im ihm alle Daten für die Initialisierung des Softwaresystems angegeben werden.
Innerhalb des Setups von Mole-Office, das automatisch beim ersten Aufruf des Programms aktiviert wird, könne folgende Eigenschaften des Benutzers festgelegt werden:

- der Name,
- der Standardaufenthaltsort, also sein Büro,
- die Standardarbeitszeit,
- die Standardmittagspause,
- die Standardkooperationsbereitschaft für sein Büro,
- das Geschäftstelefon,
- das Privattelefon,
- die elektronische Postadresse,

[63] vgl. [Pos82]

- der Systemname des Terminkalenders,
- das Zeitintervall zur Überprüfung der elektronischen Post,
- das Zeitintervall zur Aktualisierung der Systemdaten[64],
- der Interessenskontext[65],
- evtl. andere eine Kooperationsbereitschaft pro Standardaufenthaltsort,
- evtl. ein, oder mehrere benutzerdefinierte Aufenthaltsorte,
- evtl. der Pfad eines Bildes im GIF-Format und
- evtl. ein Paßwort, um auch von anderen Arbeitsplatzrechnern in der Abteilung auf Mole-Office zugreifen zu können.

Gibt der Anwender einen neuen, benutzerdefinierten Ort an, so wird dieser Ort allen Systemagenten der anderen Benutzern mitgeteilt, damit auch ihnen der neue Ort bekannt ist.

Die Angaben des Benutzers werden im Systemagenten persistent gespeichert. Zusätzlich werden sie in einer zentralen Systemdatei abgelegt, damit die anderen Mitarbeiter auf den Name des neuen Benutzers zugreifen können.
In dieser Namensdatei steht:
- der Name des Benutzers,
- der Name des System-Agents,
- der Name seines Terminkalenders und

ein Verweis auf die restlichen Daten, wie z.B. auf den Interessenskontext.

Nachdem die möglichen Informationen eines Benutzers des Mole-Office-Systems erläutert wurden, wird nun ein Sicherheitskonzept für die Handhabung der Daten eingeführt.
Das zugrundeliegende Sicherheitskonzept von Mole-Office orientiert sich einerseits an dem integrierten Sicherheitskonzept des Kalendermanagementwerkzeugs CM. Andererseits kann ein Benutzer auch persönliche Vertraulichkeitsstufen einrichten, indem er zusätzliche Informationen über die Vertraulichkeit eines Termins in seinen Kalender manuell einträgt und indem er bestimmten Benutzern eine persönliche Vertraulichkeitsstufe im Setup[66] zuordnet.
Innerhalb von CM werden folgende **Sicherheitsstufen** für Termine und Aufgaben definiert:

[64] Anmerkung: Das Zeitintervalle zur Aktualisierung der Systemdaten muß angegeben werden, da der Serverprozeß des CM nicht in der Lage ist Erinnerungsnachrichten bei eingetragenen Terminen an das Softwaresystem zu senden wenn er nicht aktiv ist.
[65] siehe Kapitel 5.1 Der substantielle Bereich
[66] siehe weiter unten, im gleichen Kapitel

1. Stufe: öffentlich bzw. „Others See Time and Text“

Trägt man Termine oder Aufgaben mit dem Attribut öffentlich in den Terminkalender ein, so können alle Benutzer sowohl den genauen Zeitpunkt des Termins, als auch die näheren Beschreibungen des eingetragenen Termins einsehen.

2. Stufe: privat bzw. „Others See Time only“

Wenn ein Benutzer einen Termin oder eine Aufgabe mit dem Attribut privat in seinen Terminkalender einträgt, so können andere Benutzer, die diesen Terminkalender nach Informationen durchsuchen, die Zeit des eingetragenen Termins sehen. Die nähere Beschreibung ist ihnen aber nicht zugänglich.

3. Stufe: geheim, bzw. „Show Nothing“

Ist ein Termin oder eine Aufgabe innerhalb des Terminkalenders als geheim eingetragen, so sieht ein anderer Benutzer, der diesen Terminkalender nach Informationen durchsucht, nicht, daß ein Termin, bzw. eine Aufgabe eingetragen ist.

Der Benutzer von CM hat desweiteren die Möglichkeit, den Zugriff auf seinen Terminkalender für beliebige Benutzer zu beschränken oder zu erweitern. Dazu kann der Eigentümer des Terminkalenders, jedem anderem Benutzer, bzw. der ganzen Welt folgende **Zugriffsbeschränkungen** auferlegen:

- „Browse“: den Terminkalender für den ausgewählten Benutzer zur Informationssuche freigeben
- „ Insert“: einem anderen Benutzer die Berechtigung für das Eintragen von neuen Terminen in den eigenen Terminkalender erteilen.
- „Delete“: einem anderen Benutzer des persönlichen Vertrauens die Möglichkeit geben, Termine aus dem eigenen Terminkalender zu löschen.

Trägt ein Anwender zusätzliche Sicherheitsinformationen in Bezug auf Aktivitäten in seinen Terminkalender und in Bezug auf Mitarbeiter im Interessenskontext, der im Setup eingerichtet werden kann, ein, so können zusätzliche **Vertraulichkeitsstufen** realisiert werden. Diese sind nur indirekt für einen anderen Mitarbeiter, d.h. nur über entsprechende Systemmeldungen, sichtbar.
Im folgenden werden drei Vertraulichkeitsstufen für Mitarbeiter und Termine angeben:

1. Stufe: „kein Vertrauen“

Wird ein anderer Benutzer als nicht vertrauenswürdig eingestuft, so erhält dieser Benutzer keinen Zugriff auf irgendwelche Ereignisse, die der lokale Benutzer auslöst. Der Watch-Agent, bzw. Ask-Agent wird in diesem Fall mit der Meldung „Benutzerstatus unbekannt, Schutzverletzung!“ an den Systemagenten zurück gesendet.
Diese Stufe wird durch ein Minuszeichen („-“) in der Termininformation im CM gekennzeichnet.

2. Stufe: „Vertrauen“

Diese Einstellung ist der Standardwert für alle Aktivitäten und Benutzer, wenn keine zusätzlichen Informationen eingetragen werden. Ein entsprechender Watch, bzw. Ask-Agent wird alle nicht vertraulichen Informationen im Rahmen des Sicherheitskonzeptes von CM erhalten.
Diese Stufe wird dadurch gekennzeichnet, daß kein Sonderzeichen im CM eingetragen wird.
3. Stufe: „großes Vertrauen“

Hat ein Benutzer großes Vertrauen zu einem anderen Benutzer, so kann der andere Benutzer als vertraulich gekennzeichnete Informationen vom dem entsprechenden Anwender erhalten. Ein entsprechender Watch-, bzw. Ask-Agent wird alle, insbesondere die als vertraulich gekennzeichneten Informationen im Rahmen des Sicherheitskonzepts von CM erhalten.
Diese Stufe wird durch ein Pluszeichen („+“) in den Termininformationen des CM’s gekennzeichnet.

5.4 Der Systemkern

Die Funktionen im Systemkern bilden die Logik für die Zusammenarbeit der anderen Bereiche, die in der Zugriffsschicht des USCM-Modells liegen. Es wird unabhängig von der externen Umgebung die Kernfunktionalität des zu erstellenden Softwaresystems erläutert.

Um die im Softwaresystem zentralen Benachrichtigungsmechanismen realisieren zu können, muß das Modell der Kooperation (siehe Kapitel 2.2.2) auf Objekte des Softwaresystems abgebildet werden. Besonders geeignet dafür ist ein **Netzmodell**, in dem die Knoten die Informationsobjekte des Systems darstellen und die Kanten die Beziehungen zwischen den Informationsobjekten beschreiben.

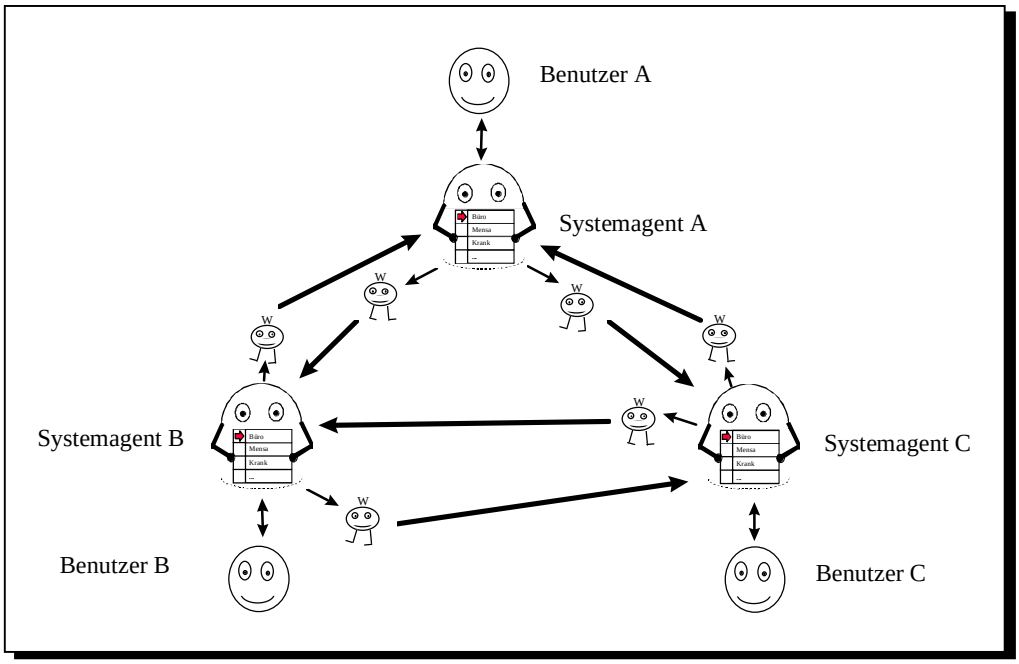

Abb.11: Das Netzmodell in Mole-Office

Die Informationen vom Benutzer werden direkt an den jeweiligen Systemagenten weitergeleitet und von ihm verwaltet. Der Systemagent sendet ggf. Nachrichten an die bei ihm registrierten Watch-Agenten, die dann wiederum ihre Informationen an ihren Systemagenten weiterleiten.
Ändert sich der Zustand eines Benutzers, so müssen davon betroffene Benutzer benachrichtigt werden. Dazu wird ein **Ereignis** im System ausgelöst, das diese Zustandsänderung dem System bekannt macht. Folgende Informationen sind in einem Ereignis enthalten:

- der Name des erzeugenden Objektes,
- der Zeitpunkt an dem das Ereignis eintrat und

- der Typ des Ereignisses.

Die Interessenkontexte[67] der Benutzer bieten die Möglichkeit zu beschreiben, an welchen Objekten, bzw. Ereignissen ein Mitarbeiter im Augenblick interessiert ist.

Wird ein Ereignis ausgelöst, sendet der Systemagent des entsprechenden Benutzers unter Wahrung der Vertraulichkeitsstufen[68] die Informationen an alle Watch-Agenten, die sich beim ihm registriert haben. Diese prüfen ihre Interessenskontexte: Werden passende Interessenskontexte gefunden, wird das Ereignis an die entsprechenden entfernten Systemagenten weitergeleitet.

Ändert z.B. Benutzer A seine subjektive Kooperationsbereitschaft, so wird eine Nachricht an den zugehörigen Systemagenten gesendet, daß sich die Kommunikationsbreitschaft des Benutzers A um 14.00 Uhr in „kontaktfreudig“ geändert hat. Dieser Systemagent sendet entsprechende Nachrichten über das Ereignis, vom Typ Benutzerereignis, an alle Watchagenten, die sich bei ihm. registriert haben. Alle Watchagenten senden nun ihrerseits entsprechende Nachrichten an ihre Systemagenten, die dann diese Nachrichten verarbeiten, den lokalen Status des Benutzers aktualisieren und an den jeweiligen Benutzer weiterreichen.

[67] siehe Kapitel 5.1 Der substantielle Bereich
[68] siehe Kapitel 5.3 Der Verwaltungsbereich

Durch eine detailliertere Untersuchung der Informationsobjekte in Mole-Office können die Ereignisse zu den grundlegenden Informationsquellen in Bezug gesetzt werden. Man erhält so den
Informations-, bzw. Datenfluß von Mole-Office:

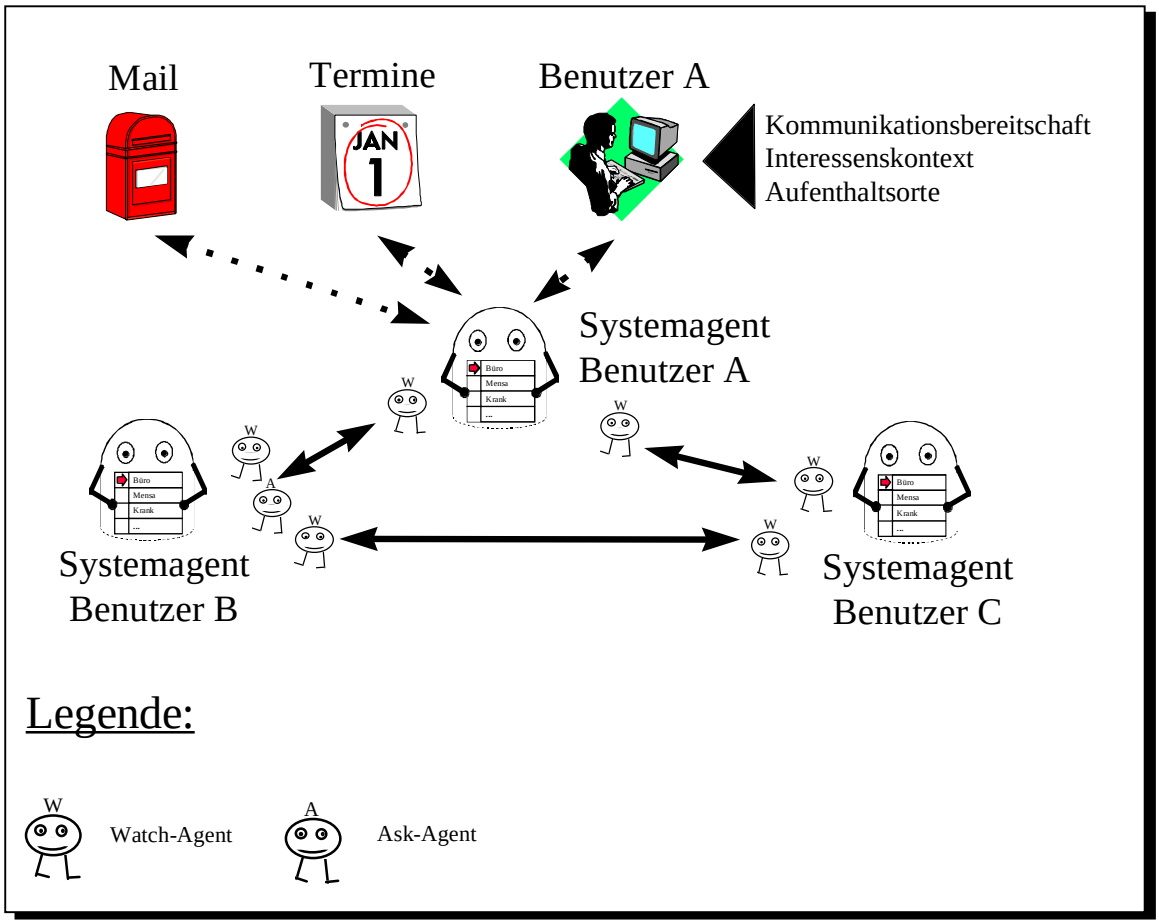

Abb.12: Informations-, bzw. Datenfluß von Mole-Office

Die grundlegenden **Informationsquellen** sind:

- Das **Mailtool**, da Mole-Office den Posteingang des jeweiligen Benutzers daraufhin überwacht, ob von Mole-Office erzeugte elektronische Post mit systemrelevanten Informationen von anderen Benutzern vorliegt.

- Das **Kalendermanagementwerkzeug**, da es ebenfalls von Mole-Office überwacht wird, um die aktuellen Termin-, bzw. Aufgabeninformationen zu erhalten.

- Die **Systemagenten der Benutzer von Mole-Office**, indem sie Informationen über die subjektive Kooperationsbereitschaft und den Aufenthaltsort des Benutzers an die entsprechenden Watch- und Ask-Agenten weiterleiten.

Der **Informations-, bzw. Datenfluß** im zu erstellenden Softwaresystem kann nun wie folgt beschrieben werden:
Die Informationen aus den grundlegenden Informationsquellen werden zum Systemagenten des jeweiligen Benutzers weitergeleitet. Dieser erzeugt entsprechende Ereignisse, die innerhalb des Agentensystems über Nachrichten an die mobilen Watch-Agenten der interessierten Benutzer weitergeleitet werden. Diese leiten eine entsprechenden Nachricht an ihren Systemagenten weiter. Der entsprechende Systemagent ändert den Zustand des Benutzers gemäß den neuen Informationen aus den Informationsquellen.
Sollen weitergehende Informationen von anderen Benutzern erfragt werden, benutzt der entsprechende Systemagent einen mobilen Ask-Agenten um die Informationen von anderen Systemagenten zu erhalten. Der Ask-Agent hat entweder die Möglichkeit zu dem entsprechenden Systemagenten zu migrieren, um dort einen lokalen Nachrichtenaustausch zu veranlassen, oder er sendet einen Remote Procedure Call an den Systemagenten[69].

Die gesamte Kommunikation von Mole-Office innerhalb des Agentensystems ist also zusätzlich zu der grundlegenden im System vorhandenen zeitlichen Steuerung[70] ereignisgesteuert und beruht auf Nachrichten zwischen den jeweiligen Agenten.

Klassifizierung der Ereignisse in Mole-Office:

Die Ereignisse in Mole-Office können anhand der grundlegenden Informationsquellen in Ereignisse aufgeteilt werden, die das Mailtool, das Kalendermanagementwerkzeug und den Benutzer des Softwaresystem selbst betreffen[71]:

Ereignisse in Bezug auf den Benutzer, bzw. Benutzerereignisse:

- Der Benutzer ändert seine subjektive Kooperationsbereitschaft für die momentane Aktivität in hoch, mittel, niedrig (**Ereignis K3, K2, und K1**).

- Der Benutzer verläßt seinen Standardaufenthaltsort und begibt sich an einen anderen Aufenthaltsort, bzw. kehrt von einem anderem Aufenthaltsort wieder an seinen Standardaufenthaltsort zurück und verändert entsprechend der definierten Standardkooperationsbereitschaft des jeweiligen Aufenthaltsortes seine Kooperationsbereitschaft (**Ereignis O1 und O2**).

[69] siehe Kapitel 5.2 Der Gebrauchsbereich
[70] siehe Kapitel 5.3 Der Verwaltungsbereich, Zeitintervalle zur Aktualisierung der Daten
[71] Anm: Die Ereignisse, die die Kooperationsbereitschaft direkt beeinflussen werden im folgenden mit dem Schriftschnitt „fett“ dargestellt.

- Der Benutzer A will genauere Informationen über den Benutzer B erhalten und startet deshalb einen Ask-Agenten (Ereignis S1).

- Ein Benutzer erweitert seinen Interessenskontext und sendet einen Watch-Agenten zu den betreffenden Systemagenten (Ereignis S2).

- Ein Benutzer definiert einen neuen Aufenthaltsort (Ereignis S3).

Die Ereignisse K1 bis K3, O1 und O2 werden benötigt, weil dadurch die Kooperationsbereitschaft des Benutzers geändert wird. Die Ereignisse S1 bis S3 dienen lediglich zur Steuerung des Verhaltens des Systemagenten.

Ereignisse in Bezug auf das Mailtool, bzw. Mail-Ereignisse

- Elektronische Post über Mole-Office versendet, bzw. erhalten (Ereignis S4).

Das Ereignis S4 ist für die Steuerung der Benutzeroberfläche notwendig, da das jeweilige Symbol des Anwenders bei Erhalt einer elektronischen Nachricht blinken soll.

Ereignisse in Bezug auf das Kalendermanagementwerkzeug, bzw. CM-Ereignisse

- Beginn eines Termins der im CM eingetragen wurde an den jeweiligen Benutzer für den Benutzerstatus melden (**Ereignis CM1**).

- Beginn eines Termins mit Priorität der im CM eingetragen wurde an den jeweiligen Benutzer für den Benutzerstatus melden (**Ereignis CM2**).

- Ende eines Termins der im CM eingetragen wurde an den jeweiligen Benutzer für den Benutzerstatus melden (**Ereignis CM3**).

- Der Benutzer erhält aufgrund einer fehlenden, bzw. zu niedrigen Vertraulichkeitsstufe keine Information von einem anderen Benutzer (Ereignis CM4).

Die Ereignisse CM1 bis CM3 sind notwendig, da sich durch Termininformationen aus dem Kalender die Kooperationsbereitschaft des Benutzers ändert. Das Ereignis CM4 ist ein Sonderfall, denn es tritt nur ein, wenn ein Benutzer Informationen von einem anderen Benutzer abfragt.

Die **Kernfunktionalität** im Softwaresystem entspricht der Zustands-, bzw. Statusverwaltung[72] des jeweiligen Benutzers in seinem Systemagenten von Mole-Office.

In welchen Zuständen sich ein Benutzer befinden kann, bzw. welche Kooperationsbereitschaft ein Benutzer nach welchen Ereignissen hat, wird durch den folgenden Automaten, bzw. durch seine Zustandstabelle, die den **ereignisgesteuerten Kontrollfluß** von Mole-Office darstellt, näher erläutert:

Zustands Nr.	**3-Tupel** (subjektive Kooperations-bereitschaft[73], Aufenthaltsort, Terminformation)	momentane **Kooperations-bereitschaft** des Benutzers	**Ereignisse**, und deren **Folgezustände**							
			K1	K2	K3	O1	O2	CM1	CM2	CM3
1.	(K1, O1, CM1)	niedrig	1	7	13	19	22	19	20	21
2.	(K1, O1, CM2)	niedrig	2	8	14	20	23	20	20	21
3.	(K1, O1, CM3)	niedrig	3	9	15	21	24	19	20	21
4.	(K1, O2, CM1)	niedrig	4	10	16	19	22	22	23	24
5.	(K1, O2, CM2)	niedrig	5	11	17	20	23	23	23	24
6.	(K1, O2, CM3)	niedrig	6	12	18	21	24	20	23	24
7.	(K2, O1, CM1)	mittel	1	7	13	19	22	19	20	21
8.	(K2, O1, CM2)	mittel	2	8	14	20	23	20	20	21
9.	(K2, O1, CM3)	mittel	3	9	15	21	24	19	20	21
10.	(K2, O2, CM1)	mittel	4	10	16	19	22	22	23	24
11.	(K2, O2, CM2)	mittel	5	11	17	20	23	23	23	24
12.	(K2, O2, CM3)	mittel	6	12	18	21	24	20	23	24
13.	(K3, O1, CM1)	hoch	1	7	13	19	22	19	20	21
14.	(K3, O1, CM2)	hoch	2	8	14	20	23	20	20	21
15.	(K3, O1, CM3)	hoch	3	9	15	21	24	19	20	21
16.	(K3, O2, CM1)	hoch	4	10	16	19	22	22	23	24
17.	(K3, O2, CM2)	hoch	5	11	17	20	23	23	23	24
18.	(K3, O2, CM3)	hoch	6	12	18	21	24	20	23	24
19.	(-, O1, CM1)	min(date&place)	1	7	13	19	22	19	20	21
20.	(-, O1, CM2)	niedrig	2	8	14	20	23	20	20	21
21.	(-, O1, CM3)	min(date&place)	3	9	15	21	24	19	20	21
22.	(-, O2, CM1)	min(date&place)	4	10	16	19	22	22	23	24
23.	(-, O2, CM2)	niedrig	5	11	17	20	23	23	23	24
24.	(-, O2, CM3)	min(date&place)	6	12	18	21	24	20	23	24

[72] siehe Kapitel 5.1 Der substantielle Bereich

[73] Anm: Das Zeichen „-“, das im folgenden verwendet wird, steht dafür, daß die vom Softwaresystem berechnete Kooperationsbereitschaft (siehe Kapitel 5.1 Der substantielle Bereich) benutzt wird, also der Benutzer keine subjektive Kooperationsbereitschaft angegeben hat.

Die einzelnen Zustände des Automaten werden durch 3-Tupel gekennzeichnet, die sich aus den im Softwaresystem möglichen Ereignissen, die die momentane Kooperationsbereitschaft des Anwenders verändern, zusammensetzen.

In diesen 3-Tupeln befinden sich an erster Stelle die Ereignisse, die sich auf die subjektive Kooperationsbereitschaft des Anwenders beziehen, d.h. der Anwender hat direkt über die Benutzerschnittstelle die subjektive Kooperationsbereitschaft der aktuellen Aktivität temporär eingestellt. Als Besonderheit wird hier das Zeichen „-“ verwendet. Es kennzeichnet, daß der Anwender die Kooperationsbereitschaft des Systems nicht geändert hat.

An zweiter Stellen des 3-Tupels stehen die Ereignisse, die den Standardaufenthaltsort des Anwenders kennzeichnen. Der Anwender kann sich hier in seinem Büro, also seinem Standardaufenthaltsort (O2) oder an einem anderen, dem Softwaresystem bekannten, Aufenthaltsort befinden (O1).

An der letzten Stelle des 3-Tupels stehen die Ereignisse, die aus Einträgen des Anwenders in seinem Kalender abgeleitet werden. Damit die Ereignisse in Bezug auf die Aktivitäten, die im Kalendermanagementwerkzeug angegeben sind automatisch erzeugt werden, ist es notwendig, daß CM eine Art von Erinnerungsnachricht an den Benutzer sendet. Leider ist der Serverprozeß des CM nicht in der Lage, diese Erinnerungsnachrichten zu senden, wenn der CM nicht aktiv ist. Insofern muß im Setup das Zeitintervall für die Überprüfung der Daten aus dem Kalender angegeben werden.

Die Zustände 1 bis 18 kennzeichnen ausschließlich die Zustände von Mole-Office, in denen der Benutzer seine subjektive Kooperationsbereitschaft temporär, also bis zum nächsten Ereignis, über die Benutzeroberfläche angegeben hat. Die momentane Kooperationsbereitschaft ist festgelegt.

Bei den Zuständen 19 bis 24 hat der Benutzer seine subjektive Kooperationsbereitschaft nicht verändert, d.h. seine momentane Kooperationsbereitschaft setzt sich aus dem Minimum der Kooperationsbereitschaft des momentanen Aufenthaltsortes und der Kooperationsbereitschaft der momentanen Aufgabe, die in seinem Kalender eingetragen ist, zusammen. Eine Ausnahme stellen hierbei die Zustände 20 und 23 dar. Hier hat der Benutzer einen Termin mit hoher Priorität in seinen Kalender eingetragen - folglich ist seine Kooperationsbereitschaft niedrig.

Im folgenden werden exemplarisch die Zustände 19 bis 24 näher erläutert:

Im **Zustand 19** hat der Benutzer einen Termin (CM1) und befindet sich außerhalb seines Büros (O1). Von diesem Zustand aus kann der Benutzer seine subjektive Kooperationsbereitschaft per Ask-Agenten in hoch, mittel oder niedrig ändern (K3, K2, oder K1) und entsprechend in die Zustände 13, 7 oder 1 springen.
Wechselt der Benutzer während eines Termins seinen Aufenthaltsort außerhalb seines Büros (O1), sendet er also einen Ask-Agenten mit entsprechenden Rauminformationen, so gelangt er wieder nach Zustand 19, wobei sich seine Kooperationsbereitschaft aus dem Minimum der Kooperationsbereitschaft des neuen Ortes und des Termins ergibt.
Kehrt der Benutzer innerhalb eines Termins in sein Büro zurück (O2), ändert sich seine Kooperationsbereitschaft entsprechend des Minimums der Kooperationsbereitschaft des Büros und der des Termins. Der neue Zustand ist nun 22.
Hat der Benutzer einen weiteren Termin innerhalb seines Kalenders eingetragen, der gerade während des anderen Termins beginnt, (CM1), so bleibt der Benutzer in Zustand 19, wobei seine Kooperationsbereitschaft entsprechend der obigen Minimumregel neu berechnet wird.
Beginnt gerade ein Termin mit Priorität (CM2), so springt der Benutzer nach Zustand 20. Die Kooperationsbereitschaft des Benutzers ist niedrig.
Beendet der Benutzer einen Termin (CM3), befindet sich aber weiterhin außerhalb seines Büros, so berechnet sich seine Kooperationsbereitschaft aus dem Minimum der Kooperationsbereitschaft des momentanen Aufenthaltsortes und der eines evtl. noch laufenden Termins. Der Zustand des Benutzers ist 21.

In **Zustand 20** befindet sich der Benutzer außerhalb seines Büro und hat einen Termin mit Priorität in seinen Kalender eingetragen. Die Kooperationsbereitschaft des Benutzers ist niedrig.
Der Benutzer kann aber trotzdem temporär seine Kooperationsbereitschaft per Ask-Agenten in hoch, mittel oder niedrig ändern (K3,K2 oder K1). Entsprechend ändert sich der Zustand des Benutzers in Zustand 14, 8 oder 2.
Wechselt während des Termins der Benutzer seinen Aufenthaltsort (O1) oder kommt er zurück in sein Büro (O2), so bleibt seine Kooperationsbereitschaft niedrig und sein Zustand ändert sich in 20, bzw. 23.
Beginnt ein neuer Termin innerhalb des aktuellen Termins (CM1, CM2), so bleibt der Benutzer mit niedriger Kooperationsbereitschaft im Zustand 20.
Endet ein Termin (CM3) und bleibt der Benutzer außerhalb seines Büros, so wechselt der Benutzer zum Zustand 21 und seine Kooperationsbereitschaft errechnet sich aus der bekannten Minimumfunktion des Ortes und eines evtl. noch vorhandenen Termins.

Hat der Benutzer einen Termin beendet und befindet sich aber noch außerhalb seines Büros befindet er sich im **Zustand 21**. Die Kooperationsbereitschaft berechnet sich aus der bekannten Minimumfunktion.
Der Benutzer kann auch hier seine subjektive Kooperationsbereitschaft in hoch, mittel oder niedrig per Ask-Agenten einstellen (K3, K2, oder K1), entsprechend springt der Benutzer in die Zustände 15, 9 oder 3.
Vor hier aus kann er seinen Aufenthaltsort in einen anderen als sein Büro wechseln (O1) oder in sein Büro zurückkehren (O2). Sein Zustand ändert sich in Zustand 21, bzw. 24, und seine Kooperationsbereitschaft wird neu berechnet.
Analog zu Zustand 19 können auch neue Termine beginnen und enden.

Die **Zustände 22 bis 24** sind direkt vergleichbar mit den Zuständen 19 bis 21, mit dem Unterschied, daß sich der Benutzer momentan in seinem Büro befindet. Entsprechend ergeben sich andere Folgezustände bei den jeweiligen Ereignissen, die aber leicht aus der Tabelle abzulesen sind. Die einzige Besonderheit ist die Tatsache, daß der Zustand 24 als Start-, bzw. Initialzustand des Softwaresystems benutzt wird, da dieser Zustand der Tatsache entspricht, daß sich der Benutzer in seinem Büro befindet und gerade einen Termin beendet hat. Falls kein Termin im Kalender eingetragen ist und der Benutzer keine subjektive Kooperationsbereitschaft angegeben hat, ist die Kooperationsbereitschaft die des Büros.

6. Implementierung

Im sechsten Teil dieser Diplomarbeit wird nun näher auf die prototypische Implementierung des Mole-Office-Systems eingegangen.
Zuerst wird dazu die Architektur des Mole-Office-Systems erläutert. Anschließend wird die Benutzeroberfläche des Systems beschrieben. Am Ende des Kapitels werden die Methoden, Aufrufe und die Kommunikation der wichtigsten Objekte im Mole-Office-System, wie z.B. die Agenten, näher erläutert[74].

6.1 Die Architektur des Mole-Office-Systems

Im Mole-Office-System wurde, ausgehend von den grundlegenden Informationsquellen[75], folgende Architektur gewählt:

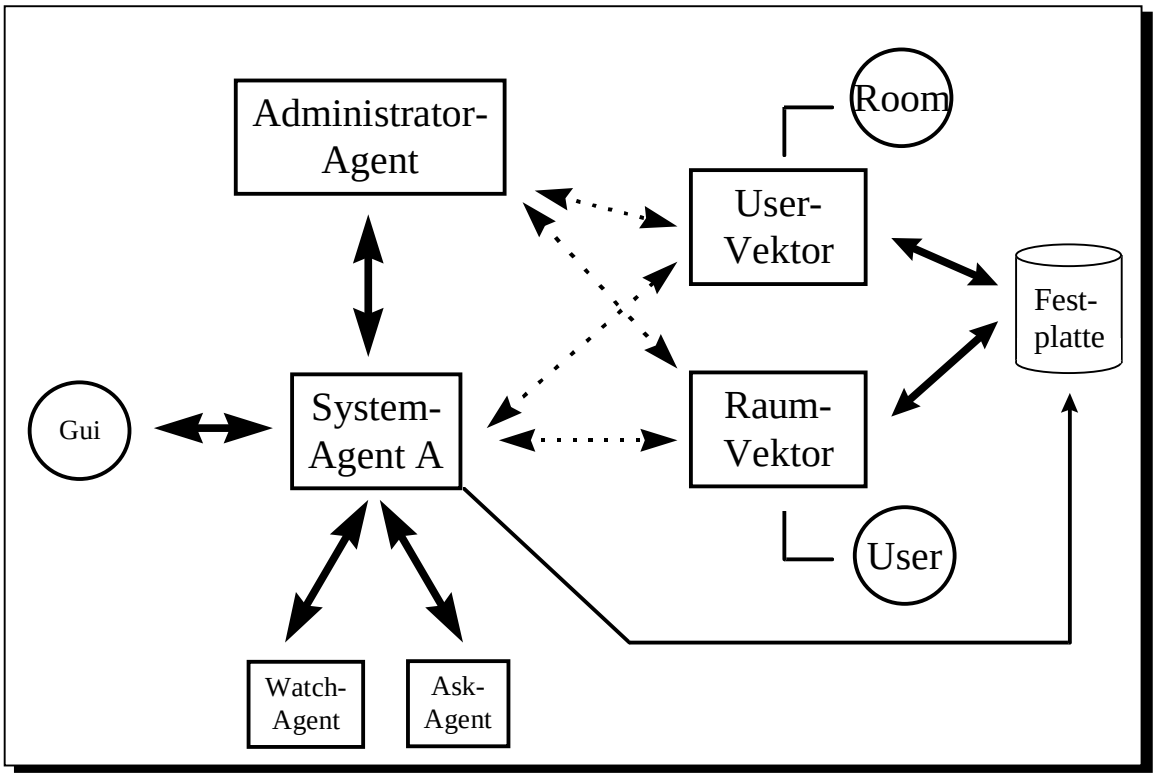

Abb.13: Architektur des Mole-Office-Systems

Die grundlegenden Objekte in Mole-Office sind die Objekte **Room** und **User**, die die Informationen in Bezug auf die Benutzer und Standardaufenthaltsorte beinhalten. Sie werden jeweils, da die Anzahl der Benutzer und Räume nicht beschränkt ist, von einem Vector (**myRoomVector, myUsersVector**) verwaltet. Diese Vectoren beinhalten auch Mechanismen, um die momentanen Daten persistent zu machen.

[74] Anmerkung: Die in den Klammer angegebenen Namen sind die real im Mole-Office-System verwendeten.
[75] siehe Kapitel 5.4 Der Systemkern

Sowohl der System-Agent als auch der Administrator-Agent des Mole-Office-Systems instanziieren jeweils Vectoren des Typs Room und User.
Alle Agenten in Mole-Office kommunizieren, nachdem die gestartet (**init, start**) wurden, über Nachrichten (**receiveMessage**), die sie sich gegenseitig senden können (**AppClient, SenderAndReceiver**).
Der **Administrator-Agent** verwaltet alle Benutzer und Räume, die neu in das Mole-Office-System eingefügt werden. Er stellt für alle Benutzer eine zentrale Informationsquelle dar, die es den ihnen ermöglicht, ihre initialen Informationen über bereits im System vorhandene Räume und Benutzer zu erhalten.
Der **System-Agent** holt sich beim Starten des Systems immer die aktuellen Informationen vom Administrator-Agenten. Er meldet auch neue Informationen, wie z.B. einen neuen Standardaufenthaltsort oder einen neuen Benutzer an den Administrator weiter, damit dieser den neuen Raum oder Benutzer allen im System vorhanden Benutzern bekannt macht. Desweiteren ist der System-Agent für die Berechnung der Kooperationsbereitschaft seines Benutzers, für das Versenden von **Ask-** und **Watch-Agenten** und für die Kommunikation mit der Benutzeroberfläche (**Gui**) verantwortlich. Damit der interne Status des System-Agenten auch nach Beendigung (**stop**) erhalten bleibt, kann der System-Agent diesen abspeichern.

Nachdem der generelle Aufbau des Mole-Office-Systems erläutert wurde, wird nun näher auf die Benutzeroberfläche von Mole-Office eingegangen.

6.2 Die Benutzeroberfläche von Mole-Office

Die als Java-Applet (**GUI**) programmierte Benutzeroberfläche von Mole-Office unterscheidet sich in einigen Details von der in Kapitel 5.1 entworfenen Benutzeroberfläche.
Die Unterschiede sind auf verschiedene Implementierungsdetails zurückzuführen auf die im folgenden näher eingegangen wird.
Das Applet, das die Benutzeroberfläche von Mole-Office darstellt, wurde mit dem Card-Layout des Abstract Windows Toolkit (AWT) von Java versehen, um die einzelnen Fenster der Benutzeroberfläche (Screen2_2, MaiLScreen, VideoScreen, SetupScreen, User-Setup, RoomSetup) schnell darstellen zu können, bzw. um größere Nachladezeiten von Appletinformationen zu vermeiden.
Das Hauptfenster (**Screen 2_2**) der Benutzeroberfläche setzt sich im wesentlichen immer noch aus einem Informations- und einem Aktionsbereich zusammen.
Der Aktionsbereich wurde im Gegensatz zum Entwurf erheblich umstrukturiert. In der obersten Zeile werden die möglichen Aktionen des Benutzers dargestellt, die ihn direkt betreffen. Dazu gehören das Setup, der CM-Aufruf-Knopf, ein Auswahlmenü mit in der Abteilung verfügbaren Aufenthaltsorten und die Möglichkeit, die momentane Kooperationsbereitschaft temporär zu ändern. Das Auswahlmenü für die Aufenthaltsorte wurde anstatt den fest codierten Raumknöpfen im Entwurf gewählt, weil dadurch erstens mehr Räume dargestellt werden und zweitens die Anzahl der Räume die zur Auswahl stehen flexibel geändert werden können.
In der zweiten Zeile des Aktionsbereichs wurden die möglichen Aktionen der Anwender in Bezug auf andere Benutzer des Mole-Office-Systems angeordnet. Dabei handelt es sich

um den Mail-Knopf, einen Videokonferenz-Knopf und einen „Update“-Knopf, mit dem man die Informationen aus dem Informationsbereich aktualisieren kann. Mittels des Mail-Knopfs ist es möglich, wie im Entwurf beschrieben, entsprechende Mails an andere Benutzer zu senden. Die Löschen- und Speicherknöpfe des Informationsbereichs wurden in das Mail-Fenster integriert.
Der Update-Knopf mußte in das Hauptfenster der Benutzeroberfläche eingefügt werden, da das Agentensystem nicht von sich aus in der Lage ist, Informationen an die Benutzeroberfläche zu senden. Die Informationen werden zwar innerhalb eines gewissen Zeitraums automatisch von der Benutzeroberfläche aktualisiert; wenn der Benutzer sich aber augenblicklich die neuesten Informationen über die anderen Benutzer anzeigen lassen will, so kann er den Update-Knopf drücken und die neusten Informationen werden sofort angezeigt.
Der Videokonferenz-Knopf wurde bereits eingefügt, weil später über Mole-Office auch Videokonferenzen abgehalten werden sollen.
Die Möglichkeit den Interessenskontext eines Benutzers anzugeben, wurde in das Setup des System verlagert, da dort noch andere Einstellungen in Bezug auf andere Benutzer vorgenommen werden können.

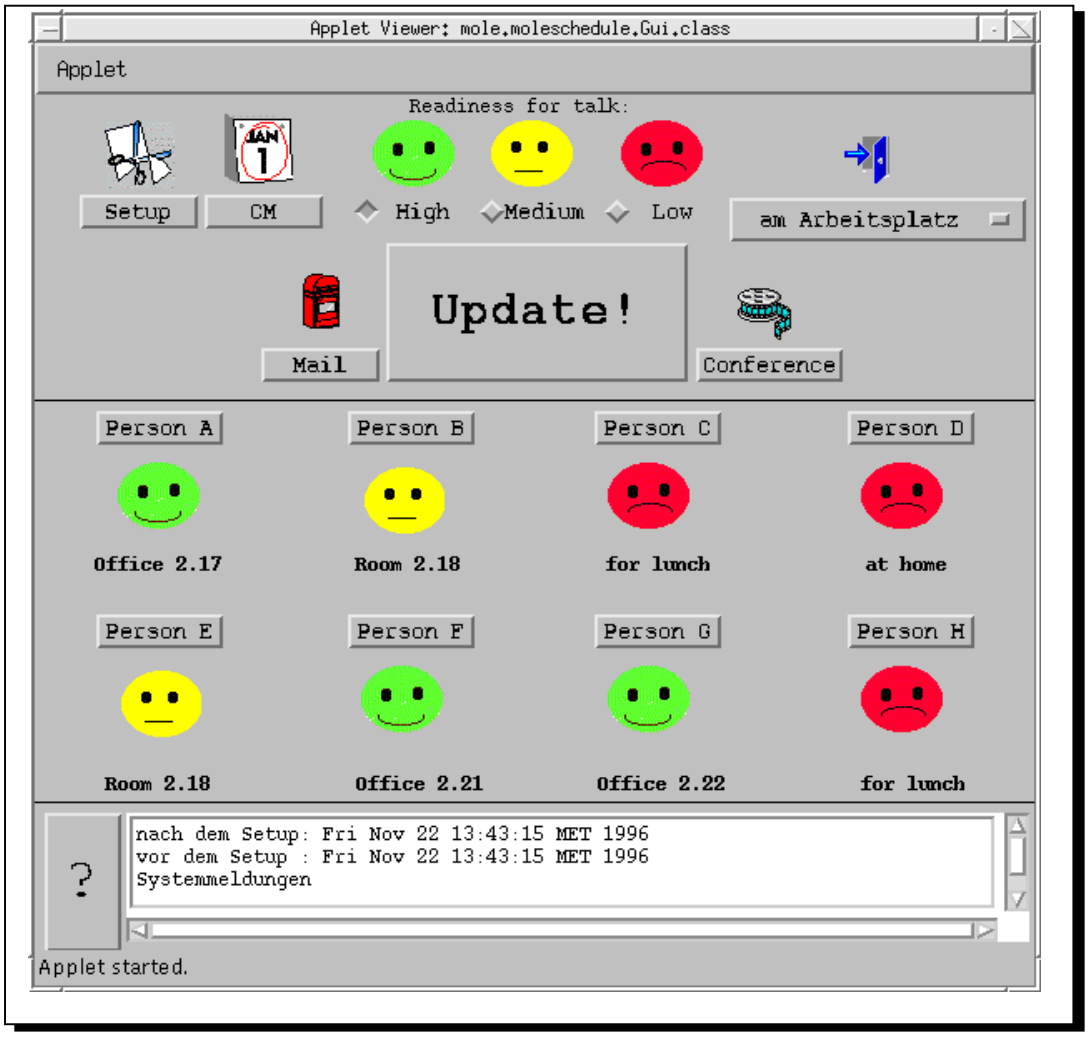

Abb.14: Das Hauptfenster der Benutzeroberfläche von Mole-Office

Im Informationsbereich wurden für jeden Benutzer, hier Person A-H, der sich im Interessenskontext des lokalen Anwenders befindet entsprechende Knöpfe mit den Namen integriert, um die speziellen Informationen über den aktuellen Benutzer, wie z.B. die Telefonnummer, abzufragen, bzw. um einen Ask-Agenten, der den Aufenthaltsort oder/und die Kooperationsbereitschaft des entsprechenden Benutzers ändert, an den Benutzer zu senden.
In der momentanen Version von Mole-Office können nur bis zu max. acht Benutzer in den Interessenskontext aufgenommen werden, da sonst die gesamte Darstellung des Applets nicht mehr auf einen Bildschirm passen würde.
Die Meldungszeile und der Hilfeknopf funktionieren, wie im Entwurf beschrieben.

Da das Setup des Mole-Office-Systems einen erheblichen Teil des Programmieraufwandes für das System ausmachte, werden nun noch die entsprechenden Setup-Fenster näher beschrieben:

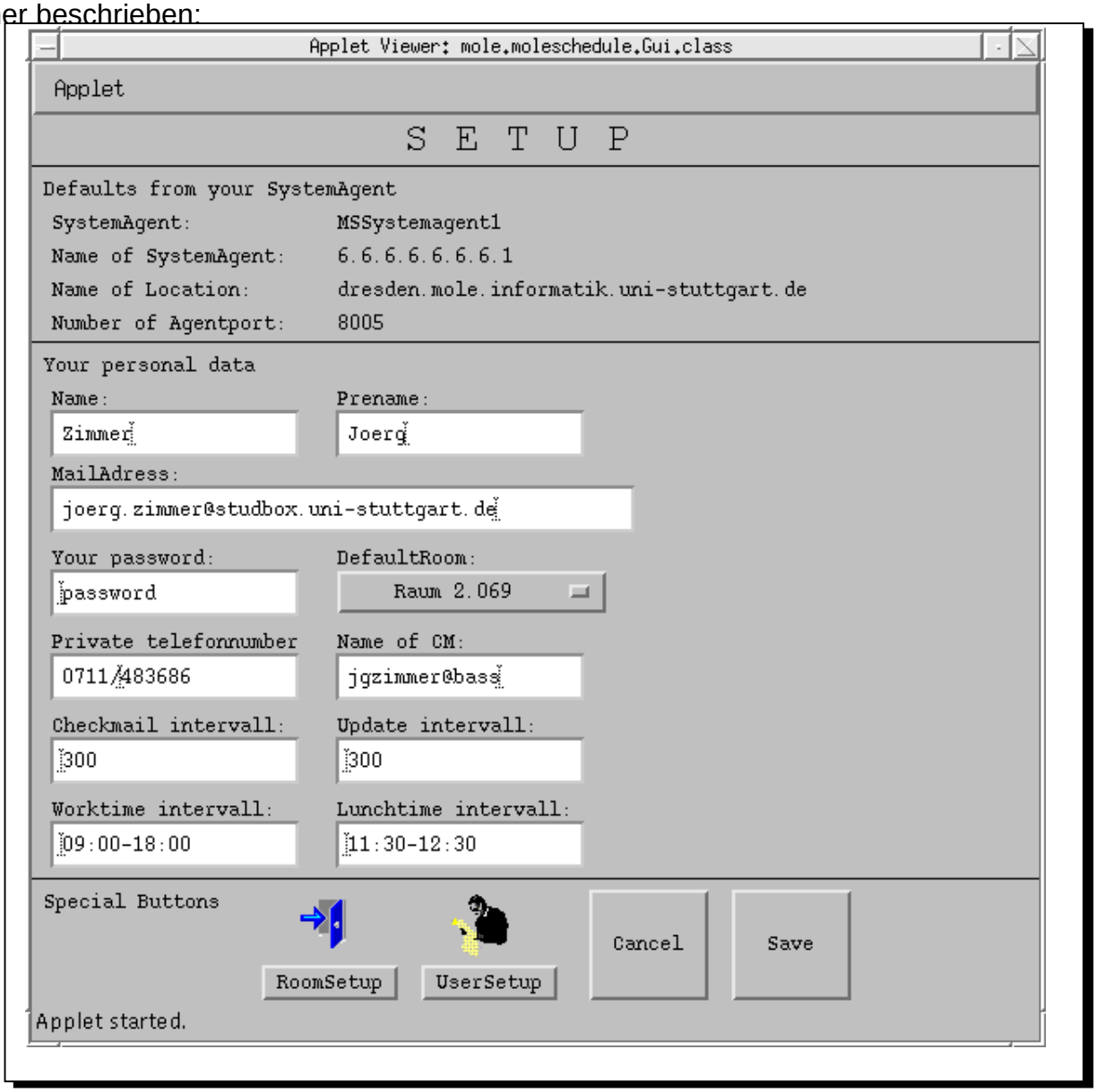

Abb.15: Das Setup-Fenster von Mole-Office

Das Setup-Fenster (**SetupScreen**) von Mole-Office kann entweder aufgerufen werden indem man den Setup-Knopf auf der Benutzeroberfläche drückt, oder automatisch beim ersten Starten des Mole-Office-Systems.
Das Setup-Fenster gliedert sich in drei unterschiedliche Teile. Im ersten Teil werden die Informationen des Systemagenten, wie die Beschreibung, der Name, die Location und der Engineport des lokalen Benutzers angezeigt. Im zweiten Teil kann der Benutzer seinen Namen, seinen Vornamen, seine Mailadresse, sein Paßwort für das Mole-Office-System, seinen Standardaufenthaltsort, seine private Telefonnummer, den Namen seines CM's, das Intervall in dem auf neue Nachrichten geprüft wird, das Update-Intervall, seine Standardarbeitszeit und seine Standardmittagszeit angeben. Das Paßwort, das man im Setup-Fenster angeben kann, ist für die Authentifikation der Ask-Agenten gedacht, damit diese den Status des Benutzers ändern können.
Im dritten Teil des Setups sind einige Spezialknöpfe vorhanden. Durch den „Cancel"-Knopf werden alle bisher vorgenommenen Änderungen wieder rückgängig gemacht und man kehrt zum Hauptfenster zurück. Drückt man den „Save"-Knopf" werden alle Änderungen abgespeichert und an den System-Agenten gesendet. Man kehrt ebenfalls zum Hauptfenster zurück.

Drückt man den „User-Setup“-Knopf, so gelangt man in das User-Setup-Fenster (**UserSetup**), in dem man die im System vorhandenen Benutzer in den eigenen Interessenskontext aufnehmen kann.

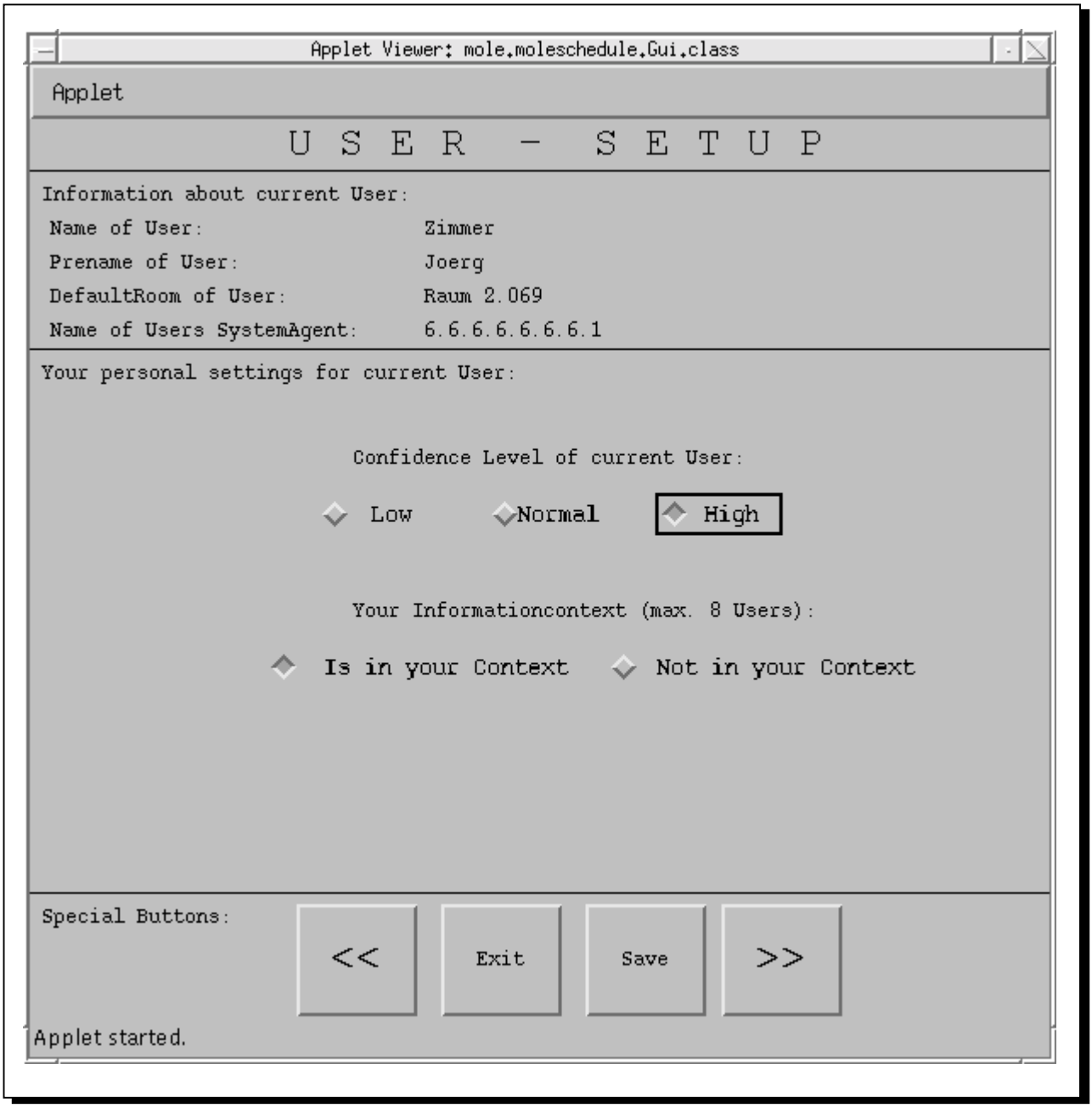

Abb.16: Das User-Setup-Fenster

Dieses Fenster ist wiederum in drei Teile unterteilt. Im ersten Teil werden allgemeine Informationen über den momentanen Benutzer angezeigt. Anhand dieser Informationen ist der Benutzer eindeutig im Mole-Office-System zu identifizieren.
Im zweiten Teil kann man seine persönlichen Konfigurationseinstellungen in Bezug auf den aktuellen Benutzer angeben. Es ist möglich für jeden Benutzer eine Vertrauensstufe

analog zu Kapitel 5.3 zu definieren. Desweiteren ist es möglich, bis zu acht Benutzer in seinen Interessenskontext aufnehmen. Diese Benutzer werden dann im Hauptfenster im Informationsbereich angezeigt. Um aktuelle Informationen zu erhalten, wird außerdem ein Watch-Agent zum entsprechenden System-Agenten gesendet. Wird der Benutzer wieder aus dem System entfernt, so wird der Watch-Agent beendet.
Im dritten Teil sind, wie auch beim Setup-Fenster, die Spezialknöpfe angeordnet. Durch den „>>“-Knopf („foreward“) kann man sich, falls vorhanden, den nächsten Benutzer anzeigen lassen. Durch den „<<“-Knopf („backwards“) entsprechend den vorherigen Benutzer.
Nur durch den „Save“-Knopf werden die Informationen abgespeichert, an den Systemagenten gesendet und das Hauptfenster der Benutzeroberfläche entsprechend angepaßt.
Drückt man den „Exit“-Knopf, so gelangt man wieder zum Setup-Fenster.
Wird im Setup-Fenster der „RoomSetup“-Knopf gedrückt, so gelangt man in das Room-Setup-Fenster (**RoomSetup**):

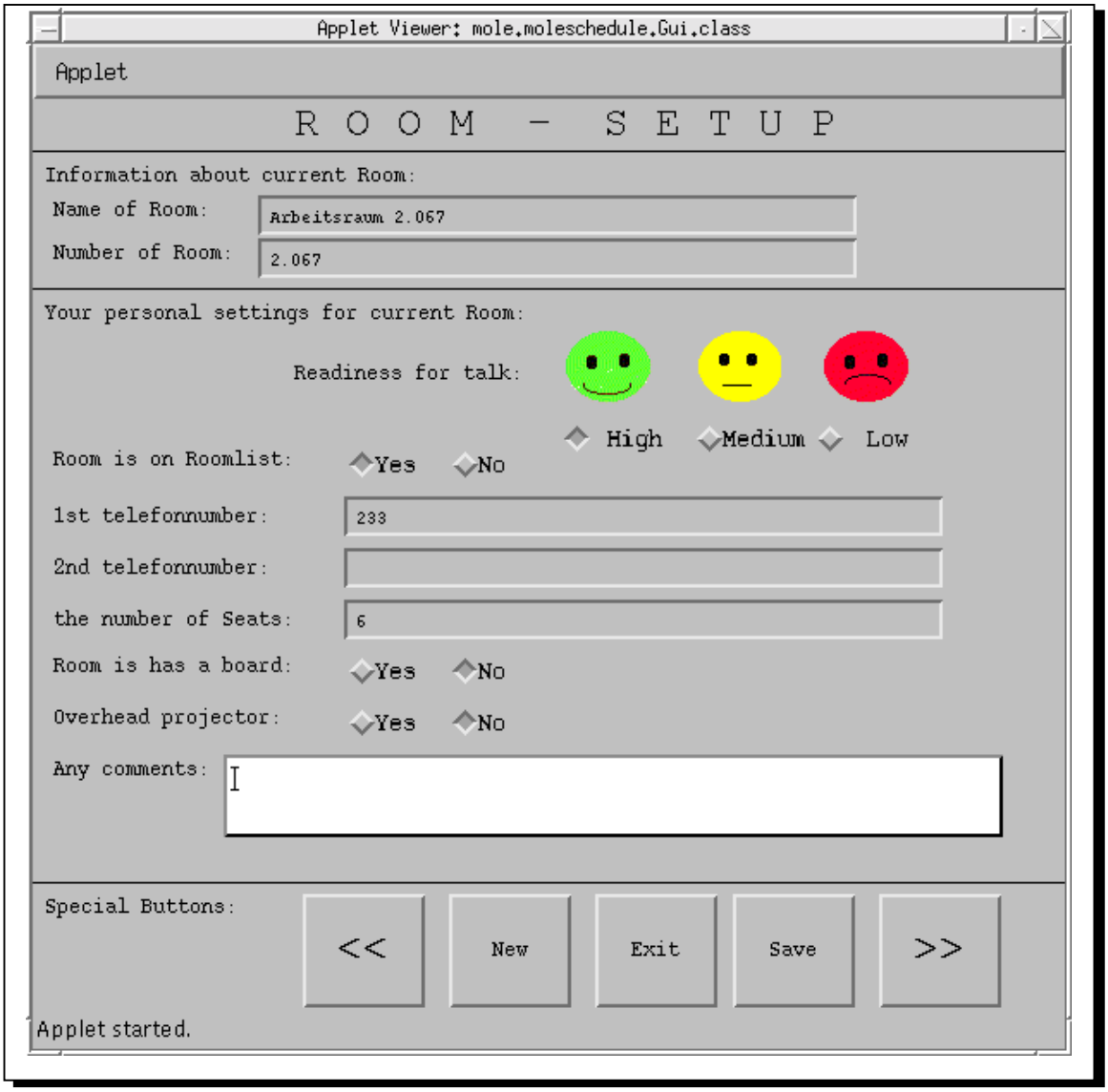

Abb.17: Das Room-Setup-Fenster

Dieses gliedert sich, wie auch die anderen Setup-Fenster, in drei Teile.
Im ersten Teil werden der Raumname und die Raumnummer, die eindeutigen Bezeichner des momentanen Raumes, angezeigt.
Im zweiten Teil kann man die Kooperationsbereitschaft pro Standardaufenthaltsraum mittels den farbigen Symbolen einstellen. Man sollte, nachdem man im Setup-Fenster seinen Standardaufenthaltsraum angegeben hat, in diesem Fenster die gewünschte Kooperationsbereitschaft im Standardaufenthaltsort angeben. Desweiteren kann man hier Kommentare in Bezug auf den aktuellen Raum angeben und auswählen, ob der Raum auf der Liste der möglichen Räume im Hauptfenster der Benutzeroberfläche erscheinen soll (OnList), oder nicht.
Weiterhin können noch die Informationen über evtl. vorhandene Telefonnummern, die Anzahl der Sitzplätze, das Vorhandensein einer Tafel oder eines Overheadprojektors angegeben, bzw. verändert werden.
Im dritten Teil des Fensters, in dem die üblichen Spezialknöpfe dargestellt werden, kann man sich auch wie beim User-Setup-Fenster mit den „foreward“, bzw. „backwards“-Knöpfen zwischen den im System vorhandenen Räumen vor- und zurück bewegen, um seine privaten Einstellungen vorzunehmen. Desweiteren existiert auch hier ein „Exit“- und ein „Save“-Knopf, mit dem man die Informationen verwerfen, bzw. abspeichern und zum Systemagenten senden kann.
Über den „New“-Knopf lassen sich neue Räume in das Softwaresystem eintragen, bzw. vorhandene Räume modifizieren. Drückt man den „New“-Knopf, so wird der Inhalt aller vorhandenen Textfelder im Room-Setup-Fenster gelöscht, und man kann einen völlig neuen Raum definieren. Ist der Raum schon im System vorhanden, so wird er entsprechend den aktuellen Änderungen des Benutzers modifiziert.

6.3 Modellierte Objekte im Mole-Office-System

Im folgenden wird nun näher auf die Methoden, Aufrufe und auf die Kommunikation der einzelnen Objekte in Mole-Office eingegangen.

6.3.1 Der Administrator-Agent (**MOAdministratorAgent**)

Der Administrator-Agent ist ein immobiler Agent, der nur einmal im gesamten Mole-Office-System vorhanden ist. Die Parameter des Agenten, wie der Agentenname (**6.6.6.6.6.6.6.0**), die Agentenlocation (**leipzig.mole.informatik.uni-stuttgart.de**) und der Agentenport (**8005**) sind fest im Mole-Office-System integriert, damit alle Benutzer, bzw. deren Agenten, wenn sie ihr lokales Mole-Office-System starten, Zugriff auf einen zen-

tralen Agenten haben, der ihnen Informationen über die anderen im System vorhandenen Benutzer und Räume gibt.
Der Administrator-Agent stellt einen zentralen „Namensdienst“ für das Mole-Office-System zur Verfügung. Dazu verwaltet er alle Benutzer des Mole-Office-Systems, die sich mittels des Setups ordnungsgemäß an das System angemeldet haben und alle Räume und Aufenthaltsorte, die, zusätzlich zu den im System bereits vorhandenen Aufenthaltsorten von den Benutzern angegeben wurden, in je einem Vektor (**User-, und Room-Vektor**).

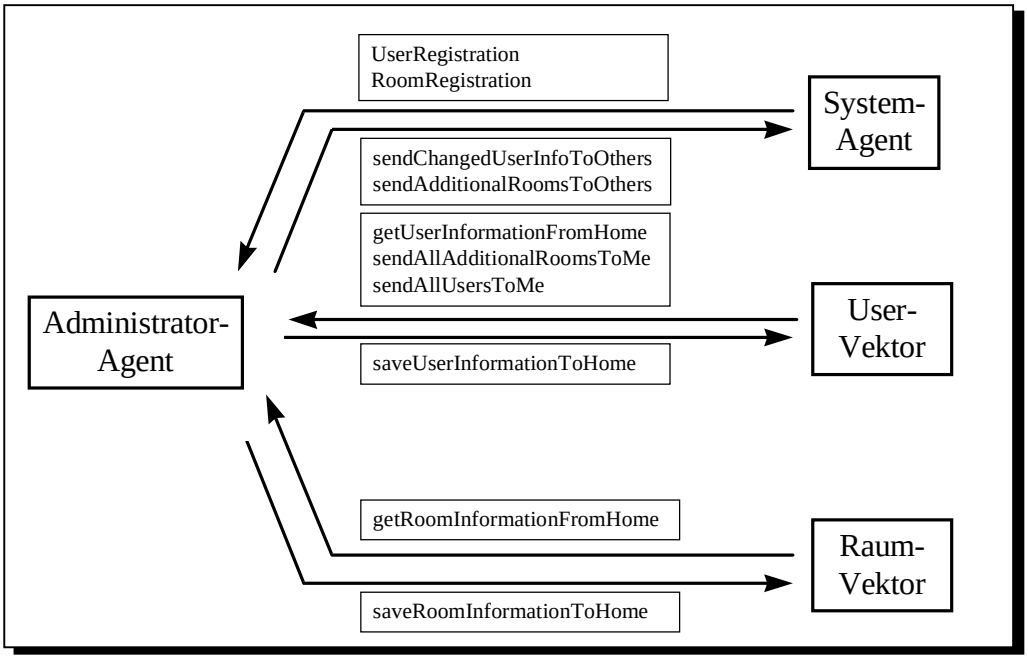

Abb.18: Methoden, Aufrufe und Kommunikation des Administrator-Agenten

Es ist einerseits möglich, dem Administrator-Agenten die eigene Benutzerinformation, wie z.B. Name, Mailadresse oder die eigene Systemagentenbeschreibung zu senden (**UserRegistration**), um die Kommunikation mit den anderen Benutzern zu ermöglichen.
Andererseits kann man auch Informationen über einen neuen Standardaufenthaltsort, wie z.B. den Namen, die Raumnummer oder die Anzahl der Sitzplätze in diesem Raum zum Adiminstrator-Agenten schicken (**RoomRegistration**).
Der Administrator-Agent speichert die neuen Informationen (**saveUserInformationToHome**, bzw. **saveRoomInformationToHome**) oder ändert seine veralteten Einträge in seiner Datenbasis und speichert danach die Änderungen. Anschließend werden alle bei ihm registrierten Benutzer von der Änderung in der globalen Datenbasis informiert (**sendChangedUserInfoToOthers**, bzw. **sendAdditionalRoomToOthers**).
Wenn ein Benutzer sein System neu startet und die Informationen über die aktuellen Benutzer und Standardaufenthaltsorte erhalten will, so können diese vom Administrator-Agenten abgefragt werden (**SendAllUsersToMe**, bzw. **sendAllAdditionalRoomsToMe**).

Wird der Administrator-Agent selbst neu gestartet, so müssen seine evtl. gespeicherten Informationen über vorhandene Räume und Benutzer wieder eingelesen werden (**getRoomInformationFromHome**, bzw. **getUserInformationFromHome**).
Wird der Agent regulär beendet, so werden die Informationen gesichert (**saveUserInformationToHome**, bzw. **saveRoomInformationToHome**).
Durch das Speichern von Informationen und durch die Möglichkeit diese wieder einzulesen, wird der gewünschte Grad an Persistenz des Administrator-Agenten erreicht.

6.3.2 Der System-Agent (**MSSystemAgent**)

Jeder Benutzer des Mole-Office-Systems hat seinen eigenen immobilen System-Agenten, der für ihn seinen Status und seine persönlichen Informationen (in Bezug auf ihn und auf andere Benutzer) sowie im System vorhandene Räume in einer lokalen Datenbasis (**User-, und Room-Vektor**) verwaltet.
Über den System-Agenten wird dem Java-Applet der Benutzeroberfläche die Möglichkeit des Zugriffs auf die Systemumgebung des Benutzers gegeben. Aus sicherheitstechnischen Gründen ist der Zugriff von Java-Applets auf das umgebende Softwaresystem untersagt. Es ist einem Java-Applet nicht möglich, auf das Filesystem zuzugreifen, um den momentanen Zustand des Benutzers abzuspeichern.

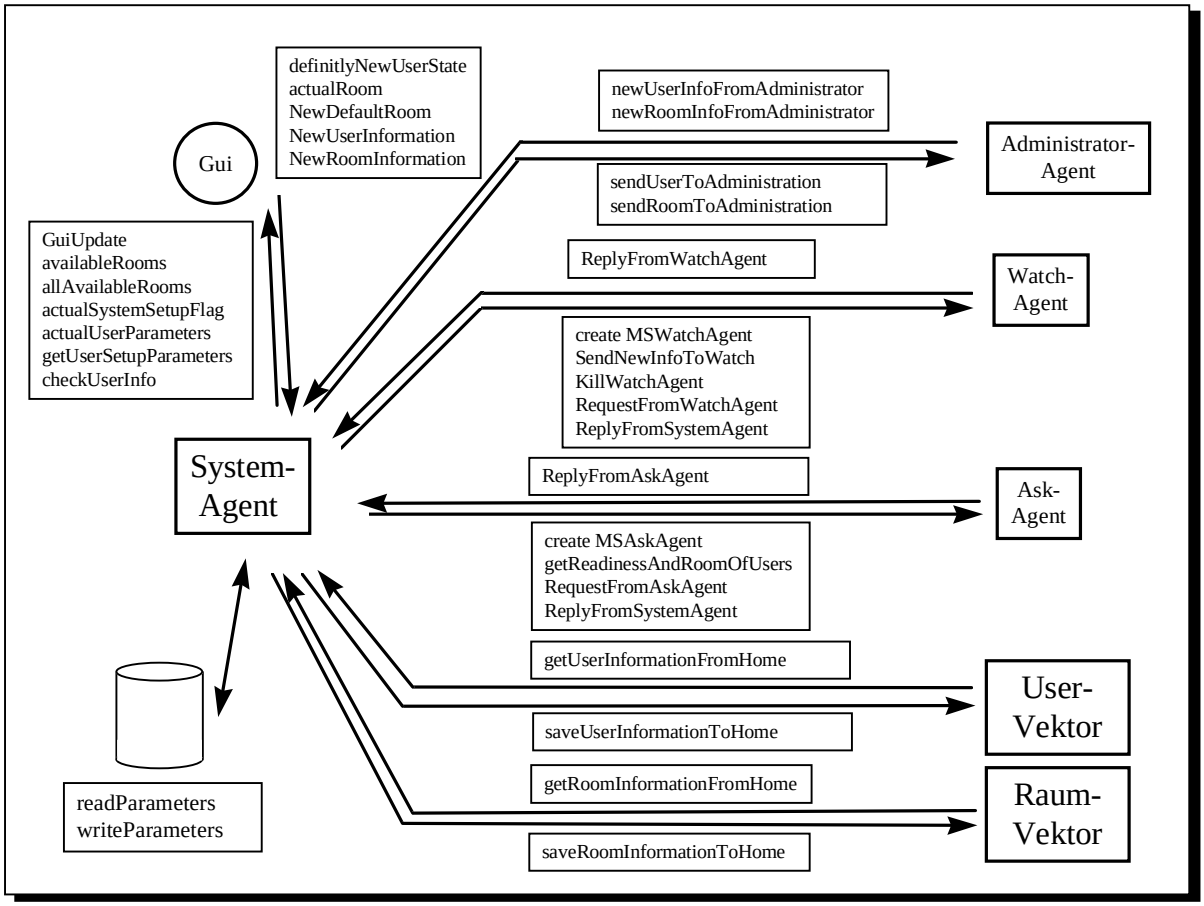

Abb19: Methoden, Aufrufe und Kommunikation des System-Agenten

Während des Startens des System-Agenten wird zuerst der aktuelle Status des im System-Agenten integrierten endlichen Automaten, der die Kooperationsbereitschaft des Benutzers berechnet falls er abgespeichert wurde, wieder eingelesen (**readParameters**). Danach werden die aktuellen Informationen über die im System vorhandenen Benutzer und Standardaufenthaltsorte eingelesen (**getRoomInformationFromHome**, bzw. **getUserInformationFromHome** und **getReadinessAndRoomOfUsers**).
Nachdem der System-Agent alle Informationen eingelesen hat, versendet er den momentanen Status an alle Watch-Agenten innerhalb dieser Location (**SendNewInfoToWatch**).

Nach diesen grundlegenden Aktionen ist der System-Agent bereit, Nachrichten anderer Agenten oder Nachrichten von der Benutzeroberfläche zu empfangen, um den Status seines Benutzers oder den der anderen Benutzer anzupassen und um Anworten zu senden.
Ändert ein Benutzer seinen Status (**definitlyNewUserState**, **actualRoom**) werden entsprechende Events erzeugt und an die registrierten Watch-Agenten weitergegeben (**SendNewInfoToWatch**).

Ändert der Benutzer seine persönlichen Einstellungen im Setup, wie z.B. seine Telefonnummer, seine Mailadresse oder seinen Standardaufenthaltsort, so wird diese Information an den Administrator-Agenten weitergeleitet (**NewDefaultRoom**, **NewUserInformation** und **sendUserToAdministration**). Der Benutzer kann aber auch einen neuen Standardaufenthaltsort im Setup angeben (**newRoomInformation**). Dieser wird dann ebenfalls an den Administrator-Agenten gesendet (**sendRoomToAdministration**). Es können auch neue Informationen über andere Benutzer und neue Räume vom Administrator-Agent an den System-Agenten gesendet werden (**newUserInfoFromAdministrator**, bzw. **newRoomInfoFromAdministrator**). Diese Informationen werden dann in die lokale Datenbasis eingetragen und bei der nächsten Anfrage auf Aktualisierung der Systemdaten an das Java-Applet weitergegeben (**GuiUpdate**).
Der System-Agent kann auch diverse Informationen über den Benutzer, über andere Benutzer und über Räume an das Java-Applet senden, so daß dieses, aufbauend auf diesen Informationen, die Benutzeroberfläche einrichten kann (**availableRooms**, **allAvailableRooms**, **actualSystemSetupFlag**, **actualUserParameters**, **getUserSetupParameters** und **checkUserInfo**).
Desweiteren hat der System-Agent die Möglichkeit, Ask-, bzw. Watch-Agenten zu erschaffen (**createMSAskAgent**, bzw. **createMSWatchAgent**), auf Anfragen der Agenten Informationen preiszugeben (**RequestFromAskAgent**, bzw. **RequestFromWatchAgent** und **ReplyFromSystemAgent**) und Informationen die von ihnen gesendet werden, in seine Datenbasis zu übernehmen (**ReplyFromAskAgent**, bzw. **ReplyFromWatchAgent**). Bei der nächsten Anfrage auf Aktualisierung der Systemdaten werden diese veränderten Informationen an das Java-Applet weitergegeben (**GuiUpdate**).

Falls sich der Interessenskontext des Benutzers ändert, kann ein Watch-Agent auch wieder gelöscht werden (**KillWatchAgent**). Ask-Agenten werden automatisch nach Erledigung ihrer Aufgabe aus dem Agentensystem entfernt.
Beim regulären Beenden des System-Agenten werden der aktuelle Status des Systemautomaten und alle Informationen über die Benutzer und Räume gesichert (**writeParameters**, **saveUserInformationToHome** und **saveRoomInformationToHome**), so daß sie beim Starten wieder zur Verfügung stehen. Somit sind alle Informationen persistent.

6.3.3 Der Ask-Agent (**MSAskAgent**)

Der Ask-Agent ist, im Gegensatz zum Administrator- und System-Agenten, ein mobiler Agent. Die grundlegende Funktionalität des Ask-Agenten besteht darin, den aktuellen Status eines anderen Benutzers unter der Wahrung von Vertraulichkeitsstufen abzufragen und an den System-Agenten weiterzuleiten.

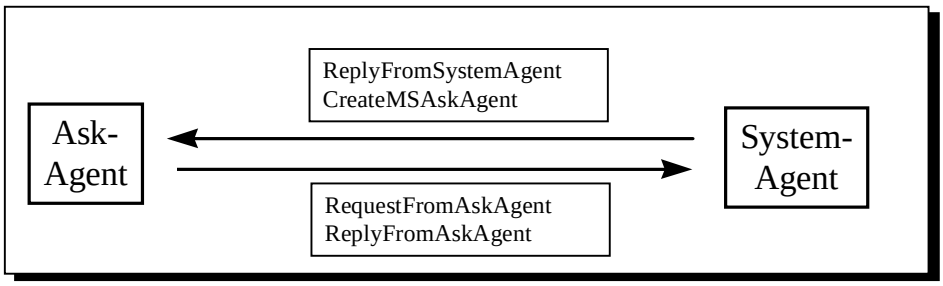

Abb.20: Methoden, Aufrufe und Kommunikation des Ask-Agenten

Dazu migriert der Ask-Agent, nachdem er erzeugt wurde (**createMSAskAgent**), zum spezifizierten Benutzer, erfragt dort den aktuellen Status, also Standardaufenthaltsort und die momentane Kooperationsbereitschaft (**RequestFromAskAgent** und **ReplyFromSystemAgent**), kehrt zum lokalen Benutzer zurück und gibt die Information an den System-Agenten weiter (**ReplyFromAskAgent**). Nachdem er diese Aufgabe erfüllt hat, wird er automatisch aus dem System entfernt.

6.3.4 Der Watch-Agent (**MSWatchAgent**)

Der Watch-Agent ist, wie auch der Ask-Agent, ein mobiler Agent. Seine Hauptaufgabe besteht darin, die Ereignisse des spezifizierten Benutzers unter der Wahrung von Vertraulichkeitsstufen zu überwachen und, falls sich der Status des Benutzers ändert, die Informationen zu seinem Besitzer zu senden, damit dieser den Status des entsprechenden Benutzers anpassen kann.

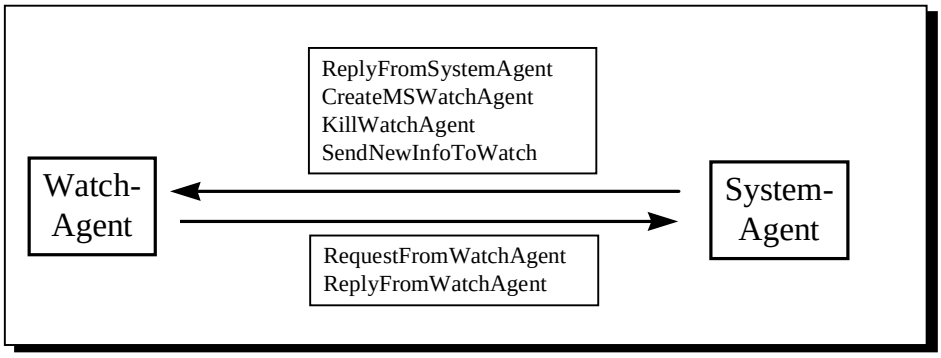

Abb.21: Methoden, Aufrufe und Kommunikation des Watch-Agenten

Nachdem der Watch-Agent erzeugt wurde (**createMSWatchAgent**), migriert er zum spezifizierten Benutzer. Dort erfragt er zuerst den aktuellen Status (**RequestFromWatchAgent**) und sendet diesen dann zum entsprechenden Besitzer zurück (**ReplyFromWatchAgent**). Danach verbleibt der Watch-Agent, im Gegensatz zum Ask-Agenten, an dieser Location, um den entsprechenden Benutzer weiterhin zu beobachten. Der Watch-Agent bleibt so lange an der Location, bis der Besitzer den entsprechenden Benutzer aus seinem Interessenskontext entfernt. Geschieht dies, wird der Watch-Agent einfach aus dem Agentensystem entfernt (**KillWatchAgent**).
Empfängt der Watch-Agent vom zu beobachtenden System-Agenten die Nachricht, daß dieser seinen Status geändert hat (**SendNewInfoToWatch**), so leitet er die Nachricht an seinen Besitzer weiter (**ReplyFromWatchAgent**), damit dieser den entsprechenden Status anpassen kann.

6.3.5 Das Benutzer-Objekt (**User**)

Im Benutzer-Objekt (**User**) werden alle einen Benutzer betreffenden Informationen gespeichert. Bei diesen Informationen handelt es sich z.B. um den Namen und Vornamen eines Benutzers, seine Mailadresse oder die Systeminformationen des System-Agenten.
Da die Anzahl der Benutzer im Mole-Office-System nicht begrenzt ist, werden alle Benutzer in einem Benutzer-Vektor (**myUsersVector**) verwaltet.

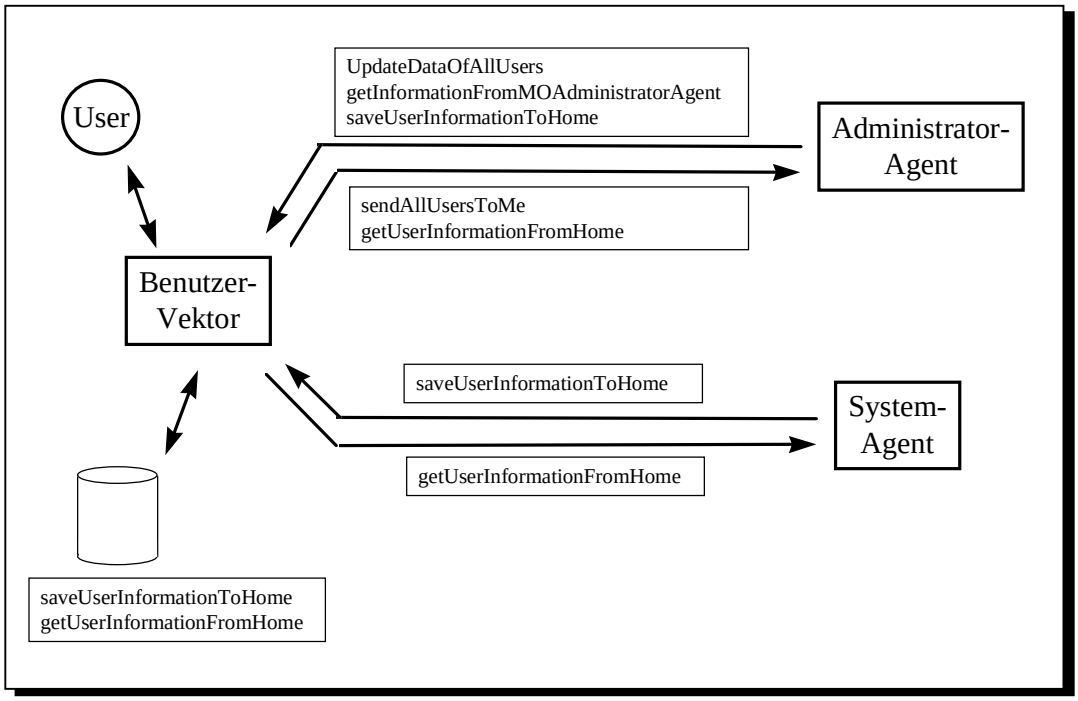

Abb:22: Methoden, Aufrufe und Kommunikation des Benutzer-Vektors
In diesem Vektor existieren die Methoden, die benötigt werden, um die Daten aus dem Benutzer-Vektor aus einem angegebenen Verzeichnis zu lesen (**getUserInformationFromHome**), bzw. in ein gegebenes Verzeichnis zu schreiben (**saveUserInformationToHome**).
Beim Starten des Mole-Office-Systems besteht die Möglichkeit zu überprüfen, ob sich bereits Benutzerinformationen im lokalen Mole-Office-System befinden. Sind Benutzerinformationen vorhanden, so werden die lokalen Informationen mit denen aus dem Administrator-Agenten verglichen und, wenn nötig, aktualisiert (**updateDataOfAllUsers**). Sind keine lokalen Benutzerinformationen vorhanden, werden alle Benutzer des Administrator-Agenten in die lokale Datenbasis eingefügt (**getUserInformationFromMOAdministrationAgent**).

6.3.6 Das Raum-Objekt (**Room**)

Im Raum-Objekt (**Room**) werden alle die Standardaufenthaltsorte betreffenden Informationen gespeichert. Diese Informationen können z.B. der Raumname, die Raumnummer und die Telefonnummer des Raumes sein.
Da die Anzahl der Räume im Mole-Office-System nicht begrenzt ist, werden alle Räume in einem Raum-Vektor (**myRoomVector**) verwaltet.

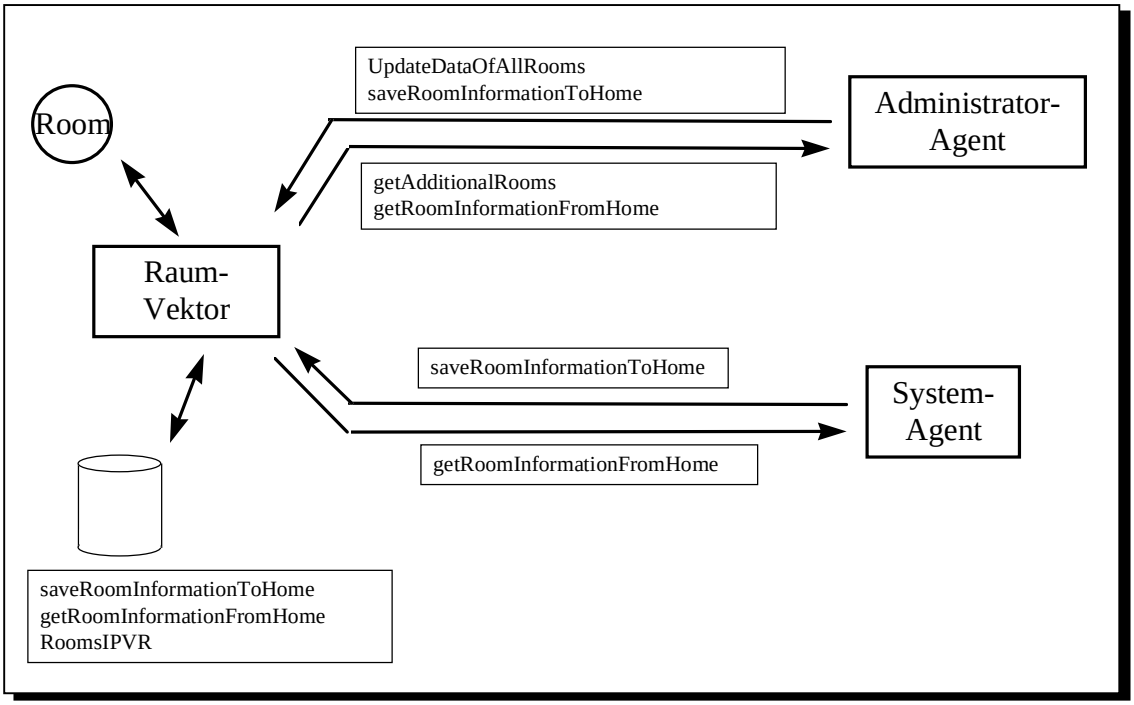

Abb.23: Methoden, Aufrufe und Kommunikation des Raum-Vektors

In diesem Vektor existieren die Methoden, die benötigt werden, um die Daten aus dem Raum-Vektor aus einem angegeben Verzeichnis zu lesen (**getRoomInformationFromHome**), bzw. in ein gegebenes Verzeichnis zu schreiben (**saveRoomInformationToHome**).
Beim Starten des Mole-Office-Systems besteht die Möglichkeit zu überprüfen, ob sich bereits Rauminformationen im lokalen Mole-Office-System befinden. Sind Rauminformationen vorhanden, so werden die lokalen Informationen mit denen aus dem Administrator-Agenten verglichen und, wenn nötig, aktualisiert (**updateDataOfAllRooms**).
Sind keine lokalen Rauminformationen vorhanden, so besteht die Möglichkeit, die Standardaufenthaltsorte der Abteilung IPVR der Universität Stuttgart in das System zu laden, da diese fest in das Mole-Office-System integriert sind (**RoomsIPVR**). Desweiteren werden alle zusätzlich von den Benutzern des Mole-Office-System eingefügten Standardaufenthaltsorte, die im Administrator-Agenten gespeichert sind in die lokale Datenbasis eingefügt (**getAdditionalRooms**).

7. Zusammenfassung

7.1 Ergebnisse

Im Rahmen dieser Diplomarbeit wurde nach eingehender Untersuchung vorhandener Koordinations- und Kooperationswerkzeuge aus dem Bereich des CSCW festgestellt, daß zur Zeit noch keine Koordinationswerkzeuge am Markt existieren, die alle für diese Diplomarbeit benötigten Informationen zur Kooperation verwalten oder bereitstellen.
Bei diesen Informationen handelt es sich um den Status und die Priorität einer Aufgabe, statische und dynamische Rauminformationen, den physikalischen Aufenthaltsort jedes Mitarbeiters in einem Team zu jedem beliebigen Zeitpunkt und den Start- und Endpunkt von Aktivitäten und Aufgaben.
Desweiteren existieren bisher keine Werkzeuge, in die verschiedene Kommunikations- und Koordinationswerkzeuge, wie elektronische Post, ein Kalendermanager oder Videokonferenzing integriert werden, die die Kooperationsbereitschaft eines Mitarbeiters in einem Team direkt am Bildschirm anzeigen können und die ein Agentensystem mit mobilen Agenten verwenden.
Daraufhin wurde das Softwaresystem Mole-Office aus der Sicht verschiedener Untersuchungsstandpunkte spezifiziert und mittels des USCM-Modells entworfen.
Anschließend wurde eine prototypische Implementierung des Softwaresystems in der Programmiersprache Java, unter der Verwendung des Agentensystems Mole, vollzogen.

Im Rahmen dieser Implementierung traten Schwierigkeiten auf, die hauptsächlich auf das Mole-Agentensystem zurückzuführen waren.
Damit vorhandene Informationen eines Benutzers beim Beenden oder Absturz des Agentensystems dem Mole-Office-System erhalten bleiben, mußte, da die Agenten von Mole nicht persistent sind, die Persistenz der Agenten selbst implementiert werden.
Desweiteren ist im Mole-Agentensystem die Kommunikation zwischen Java-Applets und Agenten momentan nur vom Applet aus möglich, d.h. das Java-Applet der Benutzeroberfläche von Mole-Office GUI muß in gewissen Zeitabständen die Information vom Systemagenten erfragen, damit es über die Aktionen im Mole-Office-System informiert wird. Es mußten, damit überhaupt eine Kommunikation in der Richtung vom Agenten zum Java-Applet stattfinden konnte, Programmteile von [Vil96] in den Programmcode übernommen werden.
Da die Kommunikation im Mole-Office-System auf dem Versenden von Nachrichten beruht, die als Daten nur aus Zeichenketten und nicht aus Objekten bestehen können, mußte weiterhin eine aufwendige Kommunikation entwickelt werden. Dafür wurde jedes zu übermittelnde Objekt zuerst auf entsprechende Zeichenketten abgebildet, dann versendet und letztendlich beim Empfänger wieder in die ursprüngliche Datenstruktur zurückverwandelt.

Im Mole-System existiert auch kein Namensdienst, über den man im Agentensystem vorhandene Agenten aufspüren könnte. Meldet man sich z.B. zum ersten Mal an das Agentensystem an, so kann keine Information über andere Benutzer in Erfahrung gebracht werden. Dazu wurde in dieser Diplomarbeit ein zentraler Namesdienst über den Administrator-Agenten eingerichtet. Da dieser Agent fest in das Mole-System integriert ist, wissen alle Agenten des Mole-Office-Systems von wo sie sich ihre initialen Informationen über andere Benutzer holen können. Der Nachteil dieser Lösung besteht darin, daß dieser Administrator einen zentralen Ausfallpunkt (engl. <single point of failure>) darstellt. D.h. wenn der zentrale Agent ausgefallen ist, bekommen neue Benutzer keine Informationen über bereits im System vorhandene Benutzer.
Eine von Mole unabhängige Schwierigkeit bei der Implementierung bestand darin, ein Setup mit in die graphische Benutzeroberfläche zu integrieren, da ein Setup eines solch großen Systems recht flexibel und somit entsprechend umfangreich ist.

7.2 Ausblick

Bei einem Softwaresystem wie Mole-Office existieren viele Erweiterungsmöglichkeiten, die den ohnehin knappen Zeitrahmen einer Diplomarbeit bei weitem sprengen würden.
Angefangen bei der Integration von Videokonferenzen, die bereits angedacht wurde, über eine neue Benutzerschnittstelle, in der die Gruppenzugehörigkeit von verschiedenen Benutzern angezeigt wird, bis hin zur Verwaltung von gemeinsamen Dokumenten.
Es könnten Bewertungsmechanismen für die Migration von Ask-Agenten integriert werden, die feststellen, ob im Augenblick eine Migration eines Agenten überhaupt sinnvoll ist oder ob nur eine Nachricht mit den entsprechenden Informationen über das Netz gesendet werden sollte[76].
Weiterhin könnten Accesslisten für Termine, deren Information nur für spezielle Mitarbeiter bestimmt ist, im Mole-Office-System verwendet werden. Darauf aufbauend würden nicht nur keine Informationen an die anderen Mitarbeiter weitergegeben, sondern es wäre auch die Möglichkeit beinhaltet, bestimmten Benutzern die gesamte Information über alle Termine vorzuenthalten, d.h. diese Agenten könnten sich weder bei dem entsprechenden Systemagenten, noch beim Kalendermanagementwerkzeug CM informieren.
Es wäre möglich, für jeden Termin eine Ausführungswahrscheinlichkeit und eine Teilnahmewahrscheinlichkeit zu definieren und zu verwenden, wobei die verschiedenen Benutzer ihre Minimalvoraussetzungen für einen gemeinsamen Termin im System angeben könnten.
Durch die Integration von Aktive Mail und weiterer u.U. modernerer Kalendermanagementwerkzeuge, könnte insgesamt auf eine breitere Basis von Daten über entsprechende Mitarbeiter zurückgeriffen werden, was die Erreichbarkeit der einzelnen Anwender erhöhen würde.

[76] vgl [Voi96]

Letztendlich könnten durch automatisches Erzeugen von HTML-Seiten, die über das Internet allgemein oder paßwortgeschützt zugänglich wären und die entsprechende Kooperationsinformationen über die einzelnen Mitarbeiter enthalten würden, die Kooperationsinformationen von Mole-Office plattformunabhängig zur Verfügung gestellt werden.

8. Anhang

8.1 Installation und Start von Mole-Office

Um Mole-Office lokal bei einem Benutzer zu installieren, müssen zur Zeit folgende Schritte durchgeführt werden:

1. Installation des Agentensystems Mole
   1.1 Locationsname z.B. dresden.mole.informatik.uni-stuttgart.de
   1.2 Engineport z.B. 8005
2. Kopieren der aktuellen Daten aus /home/jgzimmer/mole/moleschedule in ein entsprechendes lokales Verzeichnis.
3. Das MAKEFILE in diesem Verzeichnis aufrufen.
4. Im Mole-Verzeichnis die Datei MAKESFILES.JAVA so erweitern, daß ein MSSystemAgent auf der eigenen Location gestartet wird. Dies wird z.B. durch Einfügen folgender Zeilen erreicht:
   - import mole.moleschedule.*;
   - String homeDir = "/home/jgzimmer";
   - MSSystemAgent sy1 = new MSSystemAgent("MSSystemagent1",
     new AgentName(6,6,6,6,6,6,6,1), new LocationName("dresden.mole.informatik.uni-stuttgart.de"), 8005, homeDir);
   - // introduce agent sy1 into dresden
     dresden.writeObject(sy1);
     dresden.flush();
5. Im Verzeichnis Mole/CONFIG die Datei Systemclasses.dat um folgende Zeilen erweitern:

   ```
   # Mole.Moleschedule
   mole.moleschedule.MOAdministratorAgent
   mole.moleschedule.MSSystemAgent
   mole.moleschedule.myUsersVector
   mole.moleschedule.myRoomVector
   mole.moleschedule.myCmDateVector
   mole.moleschedule.SenderAndReceiver
   mole.moleschedule.ReceiveAndSend
   ```

6. Die Datei MAKEFILES.JAVA übersetzen.
7. Die Datei Gui.html entsprechend den Systemparametern von Mole einstellen
   7.1 <PARAM NAME = Description Value="MSSystemagent1">
   7.2 <PARAM NAME = AgentName Value="6.6.6.6.6.6.6.1">

7.3 <PARAM NAME = LocationName Value="dresden.mole.informatik.uni-stuttgart.de">

7.4 <PARAM NAME = AgentPort Value="8005">

8. Das Agentensystem starten
9. Gui.html z.B. im Appletviewer aufrufen.

8.2 Die momentane Bibliothek von Mole-Office

8.2.1 Modellierte Objekte im Mole-Office-System

public class: Room *// Raum-Objekt*

variables: public String number
// Raumnummer
public String name
// Raumname
public String telefonNumberA
// erste Telefonnummer
public String telefonNumberB
// zweite Telefonnummer
public String numberOfSeats
// Anzahl der Sitzplätze
public boolean board
// Ist eine Tafel vorhanden?
public boolean overhead
// Ist ein Overheadprojektor vorhanden?
public int readinessForTalk
// Kooperationsbereistschaft für diesen Raum
public boolean onList
// Soll der Raum auf der Liste der möglichen Standardaufenthalträume auftauchen
public String comment
// Kommentare in Bezug auf diesen Raum

public class: User *// Benutzer-Objekt*

variables: public boolean isEgo
// sind es die eigenen Werte?
public String name
// Name
public String prename
// Vorname
public String mailAdress
// MailAdresse
public String password
// Password für Ask-Agenten
public String telefonNumber_p
// private Telefonnummer
public String nameOfCm
// Name des CM
public String intervallOfMail
// Zeitintervall um Mail zu checken
public String intervallOfUpdate
// Zeitiuntervall für Update

public String timeOfWork
// Standard-Arbeitszeit
public String timeOfLunch
// Standard-Mittagszeit
public String homeRoom
// Default-Raum
public int readinessForTalk
// augenblickliche Kooperationsbereitschaft
public String telefonNumber_g
// augenblickliche geschäftliche Telefonnummer
public String agentDescription
// System-Agenten-Beschreibung
public String agentName
// System-Agenten-Name
public String locationName
// Locationsname des System-Agenten
public int agentPort
// Engine-Port des Location
public String room
// momentaner Aufenthaltsort
public int confidenceLevel
// Vertraunensstufe, nur bei anderen sinnvoll
public boolean informationContext
// im Interessenskontext aufgenommen
public int numberOfWatchAgents
// Anzahl der Watch-Agenten (max. 8)

public class: CmDate *// Termininformationen aus CM*
variables: public String actionBegin
// Beginn eines Termins
public String actionEnd
// Ende eines Termins
public String actionDate
// Datum eines Termins
public String actionDescription
// Beschreibung eines Termins
public boolean actionPriority
// Priorität eines Termins
public String actionRoom
// Ort eines Termins
public int actionReadinessForTalk
// subj. Kooperationsbereitschaft
public int actionConfidencelevel
// Vertraulichkeitsstufe

public class: myRoomVector *// Vektor um Raumdaten zu verwalten*

methods: public Room toRoom(Object o)
// Cast
public void saveRoomInformationToHome(String aHomeDir, myRoomVector allRoomVector)
// Rauminformation auf Festplatte schreiben
public void getRoomInformationFromHome(String aHomeDir, myRoomVector allRoomVector)
// Rauminformationen von Festplatte lesen
public void updateDataOfAllRooms(my RoomVector aRoomVector)
// neuste Informationen vom Administrator senden lassen
public void getAdditionalRooms(myRoomVector v)
// initial zusätzliche Räume von Administrator senden lassen
public void roomsIPVR(myRoomVector v)
// Alle Räume des IPVR

public class: myUsersVector *// Vektor um Benutzerdaten zu verwalten*

methods: public User toUser(Object o)
// Cast
public void saveUserInformationToHome(String aHomeDir, myUsersVector allUsersVector)
// Benutzerinformationen auf Festplatte schreiben
public boolean getUserInformationFromHome(String aHomeDir, myUsersVector allUsersVector)
// Benutzerinformationen von Festplatte lesen
public void updateDataOfAllUsers(myUsersVector aUsersVector)
// neuste Informationen vom Adminitrator holen
void getUserInformationFromMOAdministrationAgent(myUsersVector v)
// initial alle Benutzerinformationen vom Adminitrator holen

public class: myCmDateVector *// Vektor um CM-Daten zu verwalten*

method: public CmDate toDate(Object o)
// Cast

public class: MOAdministratorAgent *//Administrator-Agent*

methods: public MOAdministratorAgent()
// constructor
publicMOAdministratorAgent(String einString, AgentName, aName, LocationName h, int p, String aHomeDir)
// constructor
public void start()
// beim Starten des Agenten
public void stop()
// beim Stoppen des Agenten
public void end()
// beim Beenden des Agenten
public void heartbeat(long seconds)
// periodische Wiederholungen
public void receiveMessage(Message m)
// Nachrichten erhalten
messages: "AskAgent"
// Anfrage eines Ask-Agenten beantworten
"sendAllAdditionalRoomsToMe"
// alle zusätzlichen Orte an den entsprechenden Raum-Vektor senden
"sendAllUsersToMe"
// alle registrierten Benutzer an den entsprechenden Benutzer-Vektor senden
"UserRegistration"
// neuen BenutzerInformation vom System-Agenten erhalten
"ReplyFromAskAgent"
// Anwort eines Ask-Agenten erhalten
public void sendChangedUserInfoToOthers(String aName, String aPreName,
String aTelefonNumber_p, String aNameOfCm, String aHomeRoom, String
aAgentDescription, String aAgentName, String aLocationName, String aAgentPort)
// geänderte Benutzerinformationen an alle anderen Benutzer senden
public void createMSAskAgent(String einString, AgentName aName, LocationName home, LocationName target, AgentName SystemAgent)
// einen Ask-Agenten erschaffen

public class: MSSystemAgent *// System-Agent*

methods: public MSSystemAgent()
// constructor
public MSSystemAgent(String einString, AgentName aName, LocationName h, int p, String aHomeDir)
// constructor
public void start()
// beim Starten des Agenten
public void stop()
// beim Stoppen des Agenten
public void end()
// beim Beenden des Agenten
public void heartbeat(long seconds)
// periodische Wiederholungen (Update)
public void getReadinessAndRoomOfUsers(int aNumberOfUsers, myUsersVector v)
// Ask-Agenten für jeden Benutzer versenden, um aktuellen Status zu erhalten
public int readParameters(String aDescription, AgentName aName, LocationName lName, int aPort, String aHome)
// Status des internen Systemautomanten wieder einlesen
public void writeParameters(String aDescription,AgentName aName, LocationName lName, int aPort, String aHome, int aState)
// Status des internen Systemautomaten auf Platte schreiben
public void receiveMessage(Message m)
// Nachrichten erhalten

messages: "AskAgent"
// Ask-Agent starten
"getUserSetupParameters"
// die Benutzerparameter an UserSetup senden
"PleaseSendReadinessAndRoom"
// Status und Ort an anderen System-Agenten senden
"availableRooms"
// alle Orte mit Onlist = true an Gui senden
"allAvailableRooms"
// alle Orte mit Raumnummer an Gui senden
"definitlyNewUserState"
// neuen Benutzerstatus setzen (K1,K2,K3)
"checkUserInfo"
// aktueller Status und Ort des Benutzer an Gui senden
"actualRoom"
// ist der aktuelle Raum der Default-Ort (O1,O2)
"actualUserParameters"
// aktuelle Benutzer-Parameter an Setup-Screen senden
"newUserInfoFromAdministrator"
// neue Benutzerinformationen vom Administrator erhalten
"NewDefaultRoom"
// neuer Default-Ort des Benutzers erhalten
"NewUserInformation"
// neue Benutzerinformationen vom Setup

"ReplyFromAskAgent"
// Anwort vom Ask-Agenten erhalten
"RequestFromAskAgent"
// Anfrage eines Ask-Agenten erhalten
void sendUserToAdministration()
// die neuen Benutzerinformationen zum Administrator senden
public void newDefaultRoom(String newOffice)
// den Default-Ort des Benutzers verändern
public void createMSAskAgent(String einString, AgentName aName, LocationName home, LocationName target, AgentName SystemAgent, int Port)
// Ask-Agenten erschaffen
public int changeUserState(String systemEvent)
// den Benutzerstatus entsprechend den Ereignissen ändern
void toState1()-24()
// zum entsprechenden Zustand übergehen

public class: MSAskAgent *// Ask-Agent*
methods: public MSAskAgent()
// constructor
public MSAskAgent(String einString, AgentName aName, LocationName h LocationName t, AgentName SystemAgent, int Port, String pw)
// construtor
public MSAskAgent(String einString, AgentName aName, LocationName h, LocationName t, Long t1, Long t2)
// constructor
public void start()
// beim Starten des Agenten
public void stop()
// beim Stoppen des Agenten
public void receiveMessage(Message m)
// Nachrichten erhalten
message: "ReplyFromMSSystemAgent"
// Antwort eines System-Agenten erhalten
"ReplyFromMOAdminitratorAgent"
// Antwort des Administrator-Agenten erhalten

public class: MSWatchAgent *// Watch-Agent*
methods: public MSWatchAgent()
// constructor
public MSWatchAgent(String einString, AgentName aName, LocationNa-me h, LocationName t)
// constructor
public MSWatchAgent(String einString, AgentName aName, LocationNa-me h, LocationName t, Long t1, Long t2)
// constructor

public void start()
// beim Starten des Agenten
public void stop()
// beim Stoppen des Agenten

8.2.2 Die Benutzeroberfläche von Mole-Office

public class: Gui *// Applet der Benutzeroberfläche*
methods: public void init()
// Initialisierung
public boolean checkIfSystemSetup(String[] aSystemAgentInfo)
// Test ob das Setup beendet ist

class: screen2_2 *// Hauptfenster der GUI*
methods: Screen2_2(Gui app, Gui aParent, SetupScreen aNewSetupScreen, MailScreen aNewMailScreen, VideoScreen aNewVideoScreen, String[] aSystemAgentInfo)
// constructor
public void paint
// BegrenzungsLinien zeichen
void checkUserStateAndRoom()
// aktueller Benutzerstatus und Aufenthaltort von Systemagenten holen

class: cmButton_CLASS *// Knopf um CM zu starten*
method: cmButton_CLASS(Gui app, Screen2_2 aParent, Systemmeldungen_CLASS Systemmeldungen)
// constructor

class: raumButton_CLASS *//Choice für alle Orte die in Screen2_2 angezeigt werden*
methods: raumButton_CLASS(Gui app, Screen2_2 aParent, Systemmeldungen_CLASS Systemmeldungen, CheckboxGroup aReadiness, Checkbox1_CLASS atalkHighBox, Checkbox1_CLASS atalkMedBox, Checkbox1_CLASS atalkLowBox, String[] systemAgentInfo, String anActualUserRoom)
// constructor
public boolean action(Event evt, Object arg)
// Eventhandler
int sendRoomToSystem(String r)
// ausgewählten Ort an System-Agenten senden und den neuen Status des Benutzers erhalten
void getAvailableRooms()

// alle verfügbaren Räume vom System-Agenten senden lassen
public void updateCheckBox1(int state)
// den Status zur Checkbox1 senden

class: Checkbox1_CLASS *// Checkbox für Readiness For Talk*
methods: Checkbox1_CLASS(String a Mode, CheckboxGroup aReadiness, boolean awert, int xCoordinate, int yCoordinate, int xWidth, int yHeight, Systemmeldungen_CLASS Systemmeldungen, String[] systemAgentInfo)
// constructor
public boolean action(Event evt, Object arg)
// Eventhandler
public void newUserState(String event)
// subj. Kooperationsbereitschaft temorär verändern und an System-Agenten melden

class: setupButton_CLASS *// Knopf für das Setup*
methods: setupButton_CLASS(Gui app, Screen2_2 aParent, Systemmeldungen_CLASS Systemmeldungen)
// constructor
public boolean action(Event evt, Object arg)
// Eventhandler
public void setNewPanel(SetupScreen aPanel)
// Link auf Panel übergeben
public void switchToSetup()
// Zum SetupScreen umschalten, wenn Setup noch nicht beendet

class: videoButton_CLASS *// Knopf für spätere Videokonferenz*
methods: videoButton_CLASS(Gui app, Screen2_2 aParent, Systemmeldungen_CLASS Systemmeldungen)
// constructor
public boolean action(Event evt, Object arg)
// Eventhandler
public void setNewPanel(VideoScreen aPanel)
// Link auf Panel übergeben

class updateButton_CLASS *// Knopf für Update*
methods: updateButton_CLASS(Gui app, Screen2_2 aParent, Systemmeldungen_CLASS Systemmeldungen, String[] systemAgentInfo)
// constructor
public boolean action(Event evt, Object arg)
// Eventhandler
void updateAndAnswer()
// Gui auf aktuellen Informationsstand bringen

class: mailButton_CLASS *// Knopf für MailScreen*
methods: mailButton_CLASS(Gui app, Screen2_2 aParent,

Systemmeldungen_CLASS Systemmeldungen)
// constructor
public boolean action(Event evt, Object arg)
// Eventhandler
public void setNewPanel(MailScreen aPanel)
// Link auf Panel übergeben

class: nameButton *// allg. Knopf erschaffen, den man unbenennen kann*
method: nameButton(Gui app, Screen2_2 aParent, String Name, int xCoordinate, int yCoordinate)
// constructor

class: roomLabel *// Label in denen die Orte der Benutzer angezeigt werden*
method: roomLabel(Gui app, Screen2_2 aParent, String roomName, int xCoordinate, int yCoordinate, int xWidth, int yHeight)
// constructor

class: Systemmeldungen_CLASS *// Textfield für Systemmeldungen*
method: Systemmeldungen_CLASS(Gui app, Screen2_2 aParent)
// constructor

class: HilfeKnopf_CLASS *// Knopf für Hilfe*
method: HilfeKnopf_CLASS(Gui app, Screen2_2 aParent)
// constructor

class: coloredPicture *// um farbige Bilder in Gui einzufügen*
methods: coloredPicture(Gui app, Screen2_2 aParent, String PictureName, int xCoordinate, int yCoordinate, int xWidth, int yHeight)
// constructor
public String getFileName()
// Dateinamen holen
public void setFileName(String aFileName)
// Dateinamen setzen
public void paint(Graphics g)
// Bild zeichnen

class: picture *// um Bilder in Gui einzufügen*
methods: Picture(Gui app, Object aParent)
// constructor
public String getFileName()
// Dateinamen holen
public void setFileName(String aFileName)
// Dateinamen setzen
public void paint(Graphics g)
// Bild zeichnen

class: screen *// ein Panel-Container*

method: Screen(Gui app, Object aParent)
// constructor

public class: SetupScreen *// Setup-Fenster*
methods: SetupScreen(Gui app, Gui aParent, UserSetup aLink, RoomSetup aLinkII, String[] aSystemAgentInfoArray)
// constructor
public void paint(Graphics g)
// Trenlinien zeichnen
public void getUserParametersFromSystem(String[] systemAgentInfo)
// Benutzerparameter vom System-Agenten holen

class: CancelAndBackKnopf_CLASS *// Setup beenden und zurück zu Screen2_2*
methods: CancelAndBackKnopf_CLASS(Gui app, SetupScreen aParent, int xCoordinate, int yCoordinate, int xWidth, int yHeight, SetupTextField aNameField, SetupTextField aPreNameField, SetupTextField aMailAdressField, SetupTextField aPasswordOfUserField, SetupTextField aPrivateTelefonNumberField, SetupTextField aNameOfCmField, SetupTextField aTimeIntervallOfMail, SetupTextField aTimeIntervallOfUpdate, SetupTextField aTimeOfWorkField, SetupTextField aTimeOfLunchField, SetupRaumButton_CLASS aDefaultRoomButton, String aDefaultRoom, String[] aSystemAgentInfo)
// constructor
public boolean action(Event evt, Object arg)
// Eventhandler
void sendNewDefaultRoomToSystem(String r)
// neuen Default-Room an System-Agent senden
public void setNewPanel(Screen2_2 aPanel)
// Link auf Panel übergeben
public void getUserParametersFromSystem(String[] systemAgentInfo)
// aktuelle Benutzerparameter vom System-Agenten holen

class: SaveAndBackKnopf_CLASS *// Änderungen abspeichern und zurückkehren auf Screen2_2*
methods: SaveAndBackKnopf_CLASS (Gui app, SetupScreen aParent, int xCoordinate, int yCoordinate, int xWidth, int yHeight, SetupTextField aNameField, SetupTextField aPreNameField, SetupTextField aMailAdressField, SetupTextField aPasswordOfUserField, SetupTextField aPrivateTelefonNumberField, SetupTextField aNameOfCmField, SetupTextField aTimeIntervallOfMail, SetupTextField aTimeIntervallOfUpdate, SetupTextField aTimeOfWorkField, SetupTextField aTimeOfLunchField, String[] aSystemAgentInfo)

// constructor
public boolean action(Event evt, Object arg)
// Eventhandler
void getInformationFromTextField()
// alle Informationen vom Textfelder holen
public void setNewPanel(Screen2_2 aPanel)
// Link auf Panel übergeben

class: SetupTextField *// Textfeld für Setup*
methods: SetupTextField(Gui app, SetupScreen aParent, int xCoordinate,
int yCoordinate, int xWidth, int yHeight, String content, int alength)
// constructor
public String cutLength(String text)
// Länge der Daten aus Textfeld kürzen

class: SetupLabel *// Label für Setup*
method: SetupLabel(Gui app, SetupScreen aParent, int xCoordinate, int yCoordinate, int xWidth, int yHeight, String labelText, int fontSize)
// constructor

class: SetupRoomButton *// Knopf für RoomSetup*
methods: SetupRoomButton(Gui app, SetupScreen aParent, int xCoordinate, int yCoordinate, int xWidth, int yHeight, RoomSetup RoomLink)
// constructor
public boolean action(Event evt, Object arg)
// Eventhandler

class: SetupUserButton *// Knopf für UserSetup*
methods: SetupUserButton(Gui app, SetupScreen aParent, int xCoordinate,
int yCoordinate, int xWidth, int yHeight, UserSetup aLink)
// constructor
public boolean action(Event evt, Object arg)
// Eventhandler

class: SetupColoredPicture *// Farbbild für Setup*
methods: SetupColoredPicture(Gui app, SetupScreen aParent, String PictureName, int xCoordinate, int yCoordinate, int xWidth, int yHeight)
// constructor
public String getFileName()
// Dateiname holen
public void setFileName(String aFileName)
// Dateiname setzen
public void paint(Graphics g)
// Bild zeichnen

class: SetupRaumButton_CLASS *// Default-Raum Knopf*
methods: SetupRaumButton_CLASS(Gui app, SetupScreen aParent, int xCoordinate, int yCoordinate, int xWidth, int yHeight, String[] aSetupSystemAgentArray, String aDefaultRoom)
// constructor
public boolean action(Event evt, Object arg)
// Eventhandler
void sendNewDefaultRoomToSystem(String r)
// neuen Default-Raum zum System-Agenten senden
void getAllAvailableRooms()
// alle Räume mit Raumnummer für Default-Room auflisten

public class: UserSetup *// UserSetup-Fenster*
methods: UserSetup(Gui app, Gui aParent, String[] aSystemAgentInfoArray)
// constructor
public void paint(Graphics g)
// Trennlinien zeichnen
public void getOtherUserParametersFromSystem(String[] systemAgentInfo, int aIndexOfUser)
// die Parameter der anderen Benutzer vom System-Agenten erfragen
public void setNewPanel(SetupScreen aPanel)
// Link auf Panel übergeben

class: ExitButton *// Exit-Knopf*
methods: ExitButton(Gui app, UserSetup aParent, int xCoordinate, int yCoordinate, intxWidth, int yHeight, int aConfidenceLevel, boolean aInformationContext, int aNumberOfWatchAgent, String[] aSystemAgentInfo, String aNameOfUser, String aNameUsersMSSystemAgent)
// constructor
public boolean action(Event evt, Object arg)
// Eventhandler
public void setNewPanel(SetupScreen aPanel)
// Link auf Panel übergeben
public void getUserParametersFromSystem(String[] systemAgentInfo)
// Benutzerparameter von System-Agenten holen

class: SaveButton *// Save-Button*
methods: SaveButton((Gui app, UserSetup aParent, int xCoordinate, int yCoordinate, intxWidth, int yHeight, int aConfidenceLevel, boolean aInformationContext, int aNumberOfWatchAgent, String[] aSystemAgentInfo, String aNameOfUser, String aNameUsersMSSystemAgent)
// constructor

public boolean action(Event evt, Object arg)
// Eventhandler
void getInformationFromTextField()
// Informationen vom Textfeld holen
public void setNewPanel(Screen2_2 aPanel)
// Link auf Panel übergeben

class: SetupLabel2 *// umbenennbarer Label*
method: SetupLabel2(Gui app, UserSetup aParent, int xCoordinate, int yCoordinate, int xWidth, int yHeight, String labelText, int fontSize)
// constructor

class: Checkbox2_CLASS *// Checkbox für Vertrauensstufe*
methods: Checkbox2_CLASS(String aMode, CheckboxGroup aConfidence, boolean awert, int xCoordinate, int yCoordinate, int xWidth, int yHeight, String[] systemAgentInfo)
// constructor
public boolean action (Event evt, Object arg)
// Eventhandler

class: Checkbox3_CLASS *// Checkbox für Interessenskontext*
methods: Checkbox3_CLASS(String aMode, CheckboxGroup aInformationContext, boolean awert, int xCoordinate, int yCoordinate, int xWidth, int yHeight, String[] systemAgentInfo)
// constructor
public boolean action (Event evt, Object arg)
// Eventhandler

class: ForewardButton *// nächsten Benutzer anzeigen Knopf*
methods: ForewardButton(Gui app, UserSetup aParent, int xCoordinate, int yCoordinate, int xWidth, int yHeight, String[] aSystemAgentInfo)
// constructor
public boolean action (Event evt, Object arg)
// Eventhandler
public void getUserParametersFromSystem(String[] systemAgentInfo)
// Benutzerparameter vom System-Agenten senden lassen

class: BackwardsButton *// vorherigen Benutzer anzeigen Knopf*
methods: BackwardsButton(Gui app, UserSetup aParent, int xCoordinate, int yCoordinate, int xWidth, int yHeight, String[] aSystemAgentInfo)
// constructor
public boolean action (Event evt, Object arg)
// Eventhandler

public void getUserParametersFromSystem(String[] systemAgentInfo)
// Benutzerparameter vom System-Agenten senden lassen

public class: RoomSetup *// RaumSetup-Fenster*

methods: RoomSetup(Gui app, Gui aParent, String[] aSystemAgentInfo-Array)
// constructor
public void paint(Graphics g)
// Trennlinien zeichnen
public void setNewPanel(SetupScreen aPanel)
// Link zu Panel übergeben

class: SetupLabel3 *// umzubennenenden Label erstellen*
method: SetupLabel3(Gui app, RoomSetup aParent, int xCoordinate, int yCoordinate, int xWidth, int yHeigth, String labelText, int fontSize)
// constructor

class: ForewardIButton *// nächsten Raum anzeigen*
methods: ForewardIButton(Gui app, RoomSetup aParent, int xCoordinate,
int yCoordinate, int xWidth, int yHeight, String[] aSystemAgentInfo)
// constructor
public boolean action(Event evt, Object arg)
// Eventhandler
public void getRoomParametersFromSystem(String[] systemAgentInfo)
// Raumparameter von System-Agenten holen

class: BackwardsIButton *// vorherigen Raum anzeigen*
methods: BackwardsIButton(Gui app, RoomSetup aParent, int xCoordinate, int yCoordinate, int xWidth, int yHeight, String[] aSystemAgentInfo)
// constructor
public boolean action(Event evt, Object arg)
// Eventhandler
public void getRoomParametersFromSystem (String[] systemAgentInfo)
// Raumparameter von System-Agenten holen

class: NewButton *// neuen Raum erstellen*
methods: New Button(Gui app, RoomSetup aParent, int xCoordinate, int yCoordinate, int xWidth, int yHeight, String[] aSystemAgentInfo)
// constructor

public boolean action(Event evt, Object arg)
// Eventhandler
public void sendRoomParametersToSystem(String[] systemAgentInfo)
// Parameter des neuen Raumes an System-Agenten senden

class: RoomSetupColoredPicture *// Farbige Bilder fürs RoomSetup*
methods: RoomSetupColoredPicture(Gui app, RoomSetup aParent, StringPictureName, int xCoordinate, int yCoordinate, int xWidth,
int yHeight)
// constructor
public String getFileName()
// Dateiname holen
public void setFileName()
// Dateiname setzen
public void paint (Graphics g)
// Bild zeichnen

class: SetupTextfield2 *// modifizierbares Textfeld*
method: SetupTextfield2(Gui app, RoomSetup a Parent, int xCoordinate,
int yCoordinate, int xWidth, int yHeight, String labelText, int fontSize, boolean canEdit)
// constructor

public class: MailScreen *// Mail-Fenster*
method: MailScreen(Gui app, Gui aParent)
// constructor

public class: TestKnopf3_CLASS *// Zurück vom Mail-Fenster zu Screen2_2*
methods: TestKnopf3_CLASS(Gui app, Mail Screen aParent)
// constructor
public boolean action(Event evt, Object arg)
// Eventhandler

public class: VideoScreen *// Video-Fenster*
method: VideoScreen(Gui app, Gui aParent)
// constructor

class: TestKnopf2_CLASS // auf Screen2_2 umschalten
methods: TestKnopf2_CLASS(Gui app, VideoScreen aParent)
// constructor
public boolean action(Event evt, Object arg)
// Eventhandler

8.3 Literaturverzeichnis

[Bur96]
Burger, C.: Kooperation in verteilten Systemen.
Universität Stuttgart, Fakultät Informatik. Vorlesungsskript WS1995/96.
http://www.informatik.uni/stuttgart.de/fachschaft/kvv/kvv-ws95/veranstaltungen/KVSV/KVSV.html

[BuSe94]
Burger, C. / Sembach, F.: Ein Überblick über Verfahren zur rechnergestützten Kooperation.
Fakultätsbericht 7/1994. Fakultät Informatik. Stuttgart: 1994.
http://www.informatik.uni-stuttgart.de/ipvr/vs/publications/publications.html#1994_burger_01

[Bus93]
Busbach, U.: Kooperation und Koordination in geographisch verteilten Organisationen.
In: Groupware-Einsatz in Organisationen, GI-FG 2.0.1. Personal Computing Symposium, number 220 in GMD-Studien. (Hrsg.): Klöckner, K.
Marburg: 1993. S.81-94.

[ChMo95]
Chapman, M. / Montesi, S.: Overall Concepts and Principles of TINA. Version 1.0. 1995.
TINA Consortium: TB_MDC.018_1.0_94

[Chi92]
Chiueh, T.: Papyrus: A History-Based VLSI Design Process Management System.
PhD thesis, University of California at Berkeley: 1992.

[CM94l]
Calendar Manager. Solaris User´s Guide. Kap.5. August 1994.
aus: Online-helpfunction Solaris 2.4.

[Cri94]
Crispin, M.: Internet Message Access Protocol - Version 4.0. University of Washington. 1994.
http://www.imap.org/docs/base.html

[DeFr92]
Dermler, G. / Froitzheim, K.: JVTOS - A Reference Model for a New Multimedia Service.
In: Proceedings of the fourth IFIP Conference on High-Speed Networks. Liege: 1992.

[DiSa93]
Diaper, D. / Sanger C. (Eds.): CSCW in Practice: an Introduction and Case Studies.
London / Berlin / Heidelberg u.a.: Springer 1993.

[Dud89]
Das neue Dudenlexikon. Meyers Lexikonredaktion 1989.

[ESM96]
PVCS Version Manager 5.2 für Administrator. Nürtingen: ESM Software, 1996.

[Fon93]
Foner, L.N.: What´s An Agent? A Sociological Case Study.
Agents Memo 93-01. Agents Group, MIT Media Lab. 1993.

[FTP96I]
Cyber Agent. Intelligent Agent Technology. FTP Software 1996.
http://www.ftp.com/cyberagents

[FTP96II]
Introduction to Intelligent Agents and FTP Software CyberAgent Technology.
FTP Software 1996.
http://www.ftp.com/cyberagents

[FTP96III]
The CyberAgent Framework for Network Management. FTP Software 1996.
http://www.ftp.com/cyberagents

[FPP95]
Fuchs, L. / Pankoke-Babatz, U. / Prinz, W.: Supporting Cooperative Awareness with Local Event Mechanisms: The GroupDesk System.
In: ECSCW `95. Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work. Eds.: Marmolin, H. / Sundblad, Y. / Schmidt, K. Dordrecht / Boston / London: Kluwer Academic Publishers 1995. pp.247-262.

[Fün95]
Fünfrocken, S.: DAFID. Agentenforschung in Deutschland.
http://www.informatik.th-darmstadt.de/fuenf/work/agenten/agenten.html

[FuNe94]
Furuta, R. / Neuwirth, C. (Eds.): CSCW `94. Proceedings of the Conference on Computer Supported Cooperative Work. Chapel Hill: ACM 1994

[GeKe94]
Genesereth, M.R. / Ketchpel, S.P.: Software Agents.
In: Communications of the ACM. July 1994. Vol.37, No.7. pp.48-53.

[GKM93]
Gronbaek, K. / Kyng, M. / Morgensen, P.: CSCW Challenges: Cooperative Design in Engineering Projects.
In: Communications of the ACM. June 1993. Vol.36, No.4. pp.67-77.

[GrPa95]
Grudin, J. / Palen, L.: Why Groupware Succeeds: Discretion or Mandate?
In: ECSCW `95. Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work. Eds.: Marmolin, H. / Sundblad, Y. / Schmidt, K. Dordrecht / Boston / London: Kluwer Academic Publishers 1995. pp.263-278.

[HCK95]
Harrison, C.G. / Chess, D.M. / Kershenbaum, A.: Research Report. Mobile Agents: Are they a good idea? IBM Research Division. T.J. Watson Research Center. Almaden / T.J. Watson / Tokyo / Zurich. 1995.

[Hoh95]
Hohl, F.: Konzeption eines einfachen Agentensystems und Implementation eines Prototyps.
Universität Stuttgart, Fakultät Informatik. Diplomarbeit Nr. 1267. 1995.

[Hoh96]
Hohl, F.: Mole Alpha 1.0 Documentation.
Universität Stuttgart, Abteilung Verteilte Systeme des IPVR. 1996.

[ICAP96]
Internet Calendar Access Protocol (ICAP). 1996.
http://www.lotus.com/calendar/20fe.htm

[ICM91]
Mac Intosh, D.J. / Conry, S. E. / Meyer, R.A.: Distributed Automated Reasoning: Issues in Coordination, Cooperation, and Performance.
In: IEEE Transactions on Systems, Man, and Cybernetics. November / December 1991. Vol.21, No.6. pp.1307-1419.

[KRW90]
Karbe, B. / Ramsperger, N. / Weiss, P.: Support of cooperative work by electronic circulation folders.
In: Proceedings of the Conference on Office Information Systems. 1990. pp.109-117.

[Li96]
Li, W.: Telescript Technoloy: Mobile Agents. General Magic White Paper. General Magic 1996. http://www.genmagic.com/Telescript/Whitepapers/wp4/whitepaper-4.html

[LiMe96]
Li, W. / Messerschmitt, D.G.: Java-To-Go. Itinerative Computing Using Java. 1996.
http://ptolemy.eecs.berkley.edu/dgm/javatools/java-to-go/

[Lud89]
Ludewig, J.: Einführung in die Informatik. Skriptum Informatik I, II.
2. Auflage. Zürich: vdf 1989.

[Mae94]
Maes, P.: Agents that Reduce Work and Information Overload.
In: Communications of the ACM. July 1994. Vol.37, No.7. pp. 31-40.

[Mae95I]
Maes, P.: Artifical Life meets Entertainment: Lifelike Autonomous Agents.
In: Special Issue on New Horizons of Commercial and Industrial AI. Communications of the ACM. November 1995. Vol.38, No.11.

[Mae95II]
Maes, P.: Intelligent Software. Programs that can act independently will case the burdens that computers put on people.
In: Scientific American. September 1995. Vol.273, No.3. pp.84-86.

[MaCr90]
Malone, T.W. / Crowston, K.: What is Coordination Theory and How Can It Help Design Cooperative Work Systems?
In: CSCW `90. Proceedings of the Conference on Computer-Supported Cooperative Work.
o.O.: ACM Press 1990. pp.357-370.

[Mas54]
Maslow, A.H.: Motivation and Personality. New York: Harper & Row 1954.

[MCF94]
Mitchell, T. / Caruana, R. / Freitag. D. u.a.: Experience with a Learning Personal Assistant.
In: Communications of the ACM. July 1994. Vol.37, No.7. pp.81-91.

[MüSc92]
Mühlhäuser, M. / Schaper, J.: Project NESTOR: New Approaches to Cooperative Multimedia Authoring / Leraning.
In: Computer Assisted Leraning, fourth International Conference, ICCAL `92, Vol. 602 von Lecture Notes in Computer Science. Eds: Tomek, I. Wolfville, Nova Scotia: Springer 1992.

[Mue93]
Müller, J. (Hrsg.): Verteilte Künstliche Intelligenz. Methoden und Anwendungen.
Mannheim / Leipzig / Wien / Zürich: BI Wissenschaftsverlag 1993.

[NRZ92]
Nodine, M.H. / Ramaswamy, S. / Zdonik, S.B.: A Cooperative Transaction Model for Design Databases.
In: Database Transaction Models for Advanced Applications. Hrsg: Elmargamid, A.K.
San Mateo: Morgan Kaufmann 1992. Kap.3, S.53-85.

[Ous95]
Ousterhout, J.: Scripts and Agents: The New Software High Ground.
Invited Talk, USENIX Conference 1995.

[PYF92]
Palaniappan, M. / Yankelovich, N. / Fitzmaurice, G. u.a.: The Envoy Framework: An Open Architecture for Agents.
In: ACM Transactions on Information Systems. July 1992. Vol. 10, No.3. pp.233-264.

[PSF95]
Paulsen, V. / Syri, A. / Fuchs, L. u.a: Cooperative Awareness in the GroupDeskSystem.
In: ECSCW´95. Proceedings of the Fourth European Conference on Computer Supported Co-

operative Work. Eds.: Marmolin, H. / Sundblad, Y. / Schmidt, K.
Dordrecht / Boston / London: Kluwer Academic Publishers 1995. pp. 81-82.

[Pei95]
Peine, H.: Programmierunterstützung für Mobilcomputer durch mobile Agenten.
Treffen der GI Fachgruppe „Betriebssysteme“, 1995.

[Pei96]
Peine, H.: Das Ara Projekt. 1996.
http://www.uni-kl.de/AG-Nehmer/Ara/ara_D.html.

[Pos82]
Postel, J.B.: Simple Mail Transfer Protocol. University of Southern California. 1982.
http://www.support.psi.net/mail/smtp/smtp.html

[Pro96]
Proshare: Personal Videoconferencing System.
http://www.intelcom/proshare/PUB/smlanwp.htm
http://www.bluefin.net/~mms/video01.html

[Rie94]
Riecken, D. (Eds.): Intelligent Agents. Communications of the ACM.
July 1994. Vol.37, No.7.

[SaTo95]
Sandor, O. / Tollmar, K.: The Collaborative Desktop - An Environment for CSCW.
In: ECSCW `95. Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work. Eds.: Marmolin, H. / Sundblad, Y. / Schmidt, K.
Dordrecht / Boston / London: Kluwer Academic Publishers 1995. pp.67-68.

[Sel94]
Selker, T.: Coach: A Teaching Agent that Learns.
In: Communications of the ACM. July 1994. Vol.37, No.7. pp.92-99.

[SeRo93]
Sembach, F. / Rothermel, K.: TEATIME: Gemeinsamer Arbeitsbereich für kooperativ bearbeitete multimediale Objekte.
In: GI / ITG Arbeitstreffen Verteilte Multimedia-Systeme. o.O.: K.G. Saur, 1993. S.174-188.

[SoCh94]
Sohlenkamp, M. / Chwelos, G.: Integrating Communication, Cooperation, and Awareness: The DIVA Virtual Office Environment.
In: CSCW `94. Proceedings of the Conference on Computer Supported Cooperative Work.
Eds.: Furuta, R. / Neuwirth, C. Chapel Hill: ACM 1994. pp.331-343.

[Som92]

Sommerville, I.: Software Engineering. 4. Auflage. Wokingham / Reading / Menlo Park u.a.: Addison-Wesley 1992.

[TIR94]
Tang, J.C. / Isaacs, E.A. / Rua, M.: Supporting Distributed Groups with a Montage of Light-
weight Interactions.
In: CSCW `94. Proceedings of the Conference on Computer Supported Cooperative Work.
Eds.: Furuta, R. / Neuwirth, C. Chapel Hill: ACM 1994. pp.23-34.

[Tel95]
Teles GmbH (Hrsg.): Teles.Online-J. Version 3.09. Benutzerhandbuch. Berlin. 1995.

[UC1]
Using Calendar. CDE User´s Guide. Kap.11.
aus: Online-helpfunction Solaris 2.4.

[Vil96]
Villinger, K.: Einkaufs- und Verkaufsagenten für den elektronischen Markt.
Universität Stuttgart, Fakultät Informatik. Diplomarbeit Nr. 1403. 1996.

[Voi96]
Voigt, Th.: Entwicklung und Implementierung eines Modells zur quantitativen Beurteilung der Implementation und der Anwendung von Remote-Execution-Mechanismen.
Universität Stuttgart, Fakultät Informatik. Diplomarbeit, Abgabe 15.12.96.

[Whi94I]
White, J.E.: Telescript Technology: The Foundation for the Electronic Marketplace. General Magic White Paper. o.O. General Magic 1994.

[Whi94II]
White, J.E.: Telescript Technology. Scenes from the Electronic Marketplace. General Magic White Paper. o.O. General Magic 1994.

[WoJe95]
Wooldridge, M.J. / Jennings, N.R (Eds.).: Intelligent Agents. ECAI-94 Workshop on Agent Theories, Architectures, and Languages. Amsterdam, The Netherlands August 8-9, 1994. Proceedings. Berlin / Heidelberg / New York u.a.: Springer 1995.

[Yan95]
Yankee Group: Communication, Collaboration, Coordination: The „Three Cs“ of Workgroup Computing.
In: Yankee Watch. March 1995. Vol.3, No.3.
http://www.collabra.com

8.4 Erklärung

ERKLÄRUNG

Ich versichere, daß ich diese Arbeit selbständig verfaßt und nur die angegebenen Hilfsmittel verwendet habe.

(Jörg Zimmer)